Unifying Control and Verification of Cyber-Physical Systems (UnCoVerCPS)

WP1 Modelling and Conformance Testing

D1.3 – Report on Conformance Testing in the Development Process

---

WP1	**D1.3 – Report on Conformance Testing in the Development Process**
Authors	Alexander Rausch, Jens Oehlerking - Bosch
Short Description	This deliverable gives the collected results on conformance testing. This includes a discussion about classical model-based and UnCoVerCPS development processes, a survey on conformance testing, updated results on the automated driving use case and a procedure for automated test generation based on Bayesian optimization.
Deliverable Type	Report
Dissemination level	Public
Delivery Date	30 Dec 2018
Contributions by	Bosch, TUM, DLR
Internal review by	Maria Prandini (PoliMi), Geoff Peggman (RUR), Matthias Althoff (TUM)

Document history:

Version	**Date**	**Author/Reviewer**	**Description**
1.0	19.12.18	Alexander Rausch	Final version

Contents

1 Overview

This deliverable consolidates the results on conformance testing within UnCoVerCPS. In particular:

- we give an in-depth survey on conformance testing for cyber-physical systems to cover the large body of work in this field in Section 2,
- we discuss how conformance testing relates to tasks within the typical automotive development process in Section 3, and
- we describe the conformance testing methods that were used within UnCoVerCPS in Section 4,
- we give an update on the conformance testing for the automated driving use case (as described in Deliverable 5.2) in Section 5.

In the following, we will discuss conformance testing with respect to the UnCoVerCPS approach, i.e., online hybrid systems verification as part of the decision making of a control algorithm. However, from a conformance testing standpoint, it is not relevant whether conformant models are used in an online or an offline verification scheme. In general, conformance testing deals with providing formalized properties of models which can be reused in any verification scheme. In the following we will therefore discuss conformance testing of physical models in the general setting, which means that all results also apply to the UnCoVerCPS approach.

Model-based engineering as used in industrial practice typically does not use formal properties of physical models to show the safety of systems. Instead models are typically used in other roles. This includes simulation models for closing the loop in model-, software-, or hardware-in-the-loop testing. While such simulations can be part of a safety argument, "hard" arguments on model quality are not always used. Instead there is a much heavier reliance on data from the actual physical systems for the actual safety argument. Another use for models in classical model-based software engineering is in control, for example embedded in a feed-forward or model-predictive controller to improve control performance. Also in this setting, the models typically do not play a central role in the safety argument, which may be obtained through black-box testing or dedicated safety mechanisms that may overrule the models. This is in contrast to the UnCoVerCPS approach, where the safety argument explicitly relies on formal verification approaches applied to these models. This is the case

both for online and offline verification approaches, as the verification result is always relative to the models that are used.

This necessitates a means for arguing the model quality on which the hybrid systems verification techniques hinge. Since the argument is ultimately based on observations from the real system, this cannot be a formal verification approach, but it must be a testing approach, which we call *conformance testing*.

In the following we use the following terms:

- *conformance checking*: Given a model and a set of observations of the real system, conformance checking shows that these observations conform to the model.
- *conformance testing*: Given a model, produce a set of observations of the real system that are sufficient to argue that the model is in fact conformant to the real system. This includes the automatic selection of new test cases according to some strategy (test input selection).
- *identification of conformant models*: From a parameterized class of models, identify a conformant model which is useful for verification and/or control.

Conformance checking is rather straightforward, as it only requires checking each observation one by one with respect to some conformance notion. Conformance testing as defined above is already difficult. Firstly, it is desirable to have a good heuristic for automated test selection, leaning towards tests which challenge the model. Secondly, it is typically not clearly defined what is "sufficient to support the safety argument." This leads to test coverage arguments or statistical arguments over the set of observations. On top of this, identification of conformant models necessitates an exploration of the design space. Typically there is a tradeoff between the robustness of a conformance property and the usefulness of the model. Models permitting all kinds of behavior are conformant to a large class of observations but are not very useful for verification and control, as the results would be extremely conservative. In general, this can be seen as a multi-objective optimization problem put on top of conformance testing. Which conformant model on the resulting Pareto front is preferable depends on its intended use, including the verification algorithm to be used.

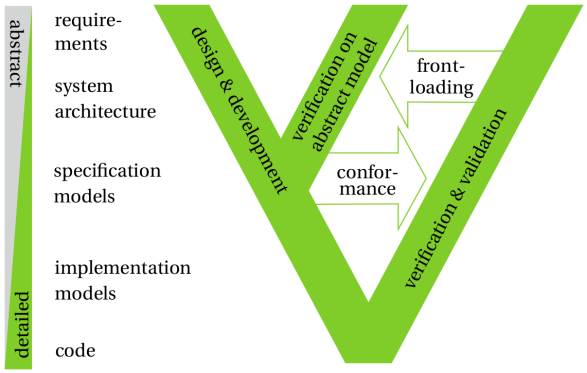


Figure 1: The V-process characterization of model-based design.

2 Conformance Testing Survey

This section gives a detailed survey on conformance testing for cyber-physical systems. This includes special considerations for the cyber-physical domain (as opposed to purely discrete systems), conformance notions, transference of properties, as well as methods from the literature for test generation. The survey was compiled alongside the work in UnCoVerCPS and forms the basis for the conformance testing methods developed within the project.

This survey is currently under review for the ACM Transactions on Cyber-Physical Systems.

2.1 Introduction

In the field of cyber-physical systems (CPS), the de facto standard for software development is model-based design. While models for software and physical components have mostly been developed separately in the past, the trend towards CPS design has led to models with mixed discrete and continuous dynamics – so called hybrid models. Discrete and continuous models have severe differences – not only in their syntax and semantics – but also in the principles used to obtain those models. The combination of discrete and continuous parts in one model is also called a hybrid system, of which the electro-mechanical brake [122] is an example.

The V-process characterizes the different development steps of a model-based design, as visualized in Figure 1. Starting with abstract requirements at the top left side of the V, the requirements are successively refined leading to a system architecture, detailed specification, and implementation models. Finally, this leads to the implemented code, shown at the bottom of the V. In the end, the requirements have to hold on the implementation. From unit test of implementation models to the abstract requirements from the beginning (advancing the top right side of the V), the implementation has to meet the expectations.

Throughout the design process, a plethora of models of the system itself or its environment are developed with a variety of purposes and on a variety of abstraction levels. In order to reduce the testing effort on the actual cyber-physical system, it is desirable to conduct as much testing as possible on these models throughout the development process. However, in order to achieve this, test results on more abstract models earlier in the process must be transferable to some extent to the actual system – the models need to be conformant to one another in some sense. However, this is still difficult to achieve in practice, due to the number of modeling tools and paradigms, the non-existence of formal semantics for many of the models, and the variety of system aspects covered in different abstractions.

From an industrial standpoint, there are various uses for conformant models on different abstraction levels: "frontloading" of testing in the design process, capturing system variability by providing different concretizations of the same abstract model, regression testing after incremental changes to the system, et cetera. The challenges that need to be overcome include the fact that formal methods in the cyber-physical domain only scale to very abstract models, the prevalence of black-box models which just exist as a compiled binary, and the necessity to eventually relate models to physical systems which can only be observed through measurements.

We call the ability to transfer properties from one model to another *transference*, and one way of achieving this is by establishing formal links between models on different levels. We call these formal links *conformance relations*. In this report, we survey different conformance relations proposed in the literature for cyber-physical systems, their transference properties and approaches for systematic testing for showing that these relations hold between given models or between models and systems.

Conformance relations for discrete systems have already been studied in detail in the literature [26, 71, 117], and can be used to support formal reasoning, e.g. showing that a system implements a specification. However, the useful abstraction to discrete systems is difficult for large classes of cyber-physical systems and properties, e.g. complex control loops, as these abstractions will typically end up being too complex to handle both by verification tools and the humans dealing with these models. For such systems continuous-time and continuous- or hybrid-valued abstractions are often both more natural and more useful.

A variety of different conformance relations for cyber-physical systems have been proposed in the literature. Aerts et al. [8] provide a brief overview about model-based testing and conformance relations for hybrid systems. However, it does not attempt to be a comprehensive survey on the subject. The goal of our survey is to provide a detailed discussion about existing notions of conformance, which does not exist in the literature, to our best knowledge. Besides

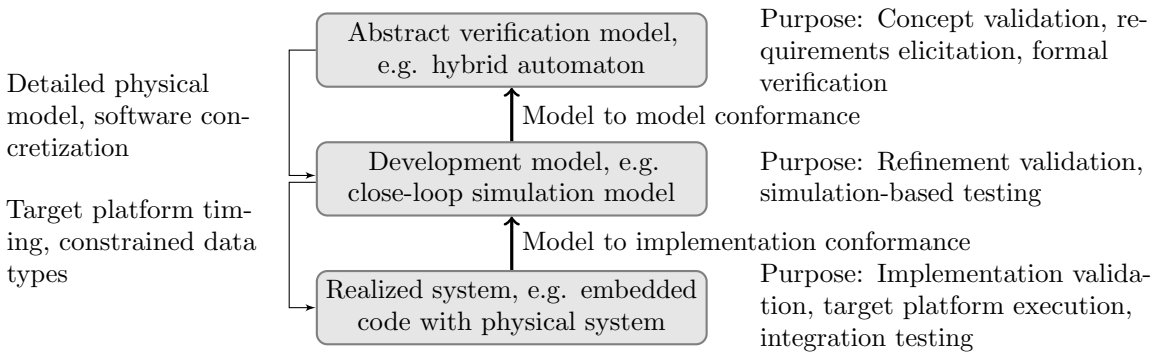


Figure 2: Different models are needed in the development process for verification and validation.

discussing the definitions and properties of conformance relations, we provide references to algorithmic approaches to check conformance and we outline the main application areas.

As such, the survey is intended for researchers, but written in a way that concepts can also be understood by practitioners. In particular, we target researchers, who are familiar with hybrid systems (e.g. modelling), but not with notions of conformance. The scope of the survey does not include stochastic models or stochastic conformance relations, both for reasons of length and the fact that conformance of stochastic models typically involves different arguments.

We first introduce the formal basis for conformance and review conformance for discrete systems in Section 2.2. In section 2.3, we discuss differences between discrete systems and hybrid systems with respect to conformance. Then, we give an overview of conformance relations and review work on conformance in the domain of cyber-physical systems in Section 2.4. We identify and study differences between several relevant conformance notions and give some guidance on how to select the right conformance relation for a given use case in Section 2.5. Finally, we discuss important aspects of input selection approaches for conformance testing in Section 2.6 and conclude in Section 2.7.

2.2 Preliminaries

At the beginning of the development process, we want to model software and physical systems by a simple model, e.g., hybrid automata as a modelling formalism, on which properties can be verified. Subsequently, the developed software is refined using simulation tools and later for implementation, e.g., on an electronic control unit. In every step, implementation aspects such as scheduling of computations, quantization of variables, and sensor/actuator inaccuracies are added as shown in Figure 2.

We consider $inputs(S)$ as the set of piecewise continuous input functions of system S. For modelling evolutions of the system, we follow Dang [41]. We define a state of a hybrid system

as a tuple (q, x), which combines the discrete location q and the continuous state $x \in \mathbb{R}^n$. A state trace x of the system is the sequence

$$x = (q_0, x_0)(q_1, x_1)\dots$$

with discrete states q_i and continuous functions x_i mapping time intervals $[t_i, t_{i+1}]$ with $t_i \leq t_{i+1}$ to continuous states. As observable, we consider output states, which are projections of states via the output map *out*. An output trace τ is the mapping of the state trace x onto the observable output space via the map *out*:

$$\forall i\ \forall t \in [t_i, t_{i+1}) : \quad \tau(t) = out(q_i, x_i(t)).$$

The set of all possible state traces of a system S under a given input function $u(.)$ is denoted by $straces(S, u(.))$. The set of all possible output traces of a system S under a given input function $u(.)$ is denoted by $otraces(S, u(.))$. Note that non-determinism in continuous dynamics can be modelled using non-deterministic inputs and using non-deterministic differential equations. Without loss of generality, we assume that non-determinism is not modelled with non-deterministic inputs, but with differential inclusions [120]. In the hybrid systems community, Zeno[1] behavior is often assumed to be absent from the model, or, if this is not possible, Zeno runs are not considered valid traces. Since Zeno behavior is a concept that cannot be observed in the physical system, it is also often of little use in the models. We assume in this report that all traces are non-Zeno.

Contrary to a (white box) abstract model with formalized differential equations, a (black box) implementation typically can only be measured via its input/output behaviour. Our concern is whether two systems on different abstraction levels in Figure 2 conform in a way that properties of one system also hold in the other one. A formal definition of the notion of conformance removes ambiguity and enables us to prove transference of properties. Conformance should be defined as permissive as possible to relate many systems, yet as strong as necessary to transfer the properties of interest.

Inspired by Tretmans [131], this can be formalized in the following way: We are given a specification *spec* and two systems, an abstract system S_A and an implementation system S_I. A specification *spec* is a property the system should have and describes a set of correct (input to) output behaviours. A system S_A is correct with respect to *spec*, which we write as $S_A \models spec$, if the output behaviour of S_A is a subset of the correct output behaviour of *spec* (for the same inputs). A useful conformance relation `conf` between systems S_I and S_A implies

[1] Infinitely many discrete transitions in finite time.

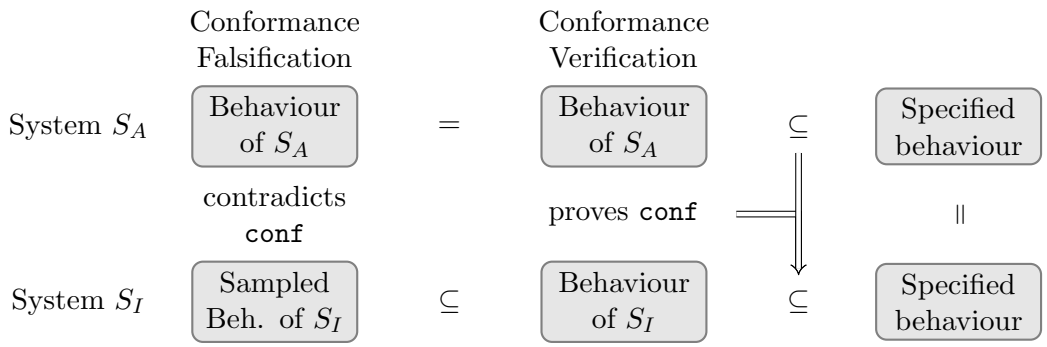


Figure 3: Conformance Overview

transference for related systems, which is that all specified properties transfer from S_A to S_I

$$S_I \,\texttt{conf}\, S_A \quad \wedge \quad S_A \models spec \qquad \Longrightarrow \qquad S_I \models spec\,.$$

There exist different names for conformance relations, such as implementation relation [26] and refinement relation [18]. If a conformance relation holds, S_A is called an abstraction of S_I and S_I is called a refinement of S_A [26]. For a conformance relation, it is important what type of properties are transfered. For the class of properties which are transfered, the conformance relation can also be called a property-preserving relation [29].

The presented formal definition of conformance enables sound reasoning: We can prove that systems are conformant (*conformance verification*) or we can find a counter-example (trace) which shows conformance cannot hold (*conformance falsification*). For conformance verification shown in Figure 3, we have to prove that all behaviours of the two systems are conformant. This is typically only possible for well-defined white box models or for measurements with further assumptions on the system behaviour, because otherwise we cannot prove that we have checked every possible behaviour. On the other hand, conformance falsification only needs sampled behaviour of S_I which does not have a corresponding equivalent in the abstract model, to prove that conformance does not hold, see Figure 3. If we use additional measurements to check conformance and do not find a counter-example, this increases the confidence that the considered systems are conformant, but does not provide a formal proof. Although it is incomplete, conformance is tested in practice with finitely many tests, which are selected to obtain a high confidence that most (relevant) behaviour has been checked. Such a conformance testing approach should effectively check conformance on already available development artefacts, e.g. abstract models built early in the development process. It should select inputs to the systems such that it fails fast, exposes relevant behavior, and stops based on an interpretable test-end criterion.

In this paper, we consider different classes of properties. A *safety property* requires traces

to satisfy a propositional formula at every point in time. An example is freedom of collisions of an autonomous vehicle, which demands no intersection with the occupancy of other traffic participants. Furthermore, the class of linear-time properties specify desired discrete traces of a discrete system [26, Section 3.2.3]. With Linear Temporal Logic (LTL) [26, Section 5], a set of desired traces can be specified. This is done by combining propositional formulas on states with temporal operators, such as *always* and *eventually*. Computational Tree Logic (CTL) [26] specifies desired computational trees of a discrete system using path quantifiers. LTL and CTL do not include each other, a discussion of the differences can be found in [26, Section 6.1]. For timed systems, there are timed versions of LTL and CTL, called timed LTL (TLTL) and timed CTL (TCTL) [26, 111, Section 9.2], which add clock constraints to the propositions on the discrete state. Metric Temporal Logic (MTL) is similar to TLTL, but defined on Boolean traces over continous time [15]. Furthermore, MTL has been extended to Signal Temporal Logic (STL) [89], which maps propositions on continuous states to Boolean traces and thus, can be used to define temporal properties for hybrid systems.

2.3 Principles of Discrete and Timed Conformance, and Concerns of CPS

The main idea of conformance relations are that the inner structure and state of the system is not relevant, as long as the behaviour on the output is similar. Conformance relations for discrete and timed systems are typically used as a foundation for conformance of hybrid systems. Therefore, we give a brief overview of discrete and timed conformance relations. Then, we discuss challenges wrt. hybrid systems.

Discrete systems can be modelled as transition systems, which produce a sequence of discrete events. *Trace inclusion* between two discrete systems S_I and S_A holds, if all traces of S_I are also traces of S_A. Trace inclusion transfers LTL properties for systems without terminal states [26, Theorem 3.15]. Another relation is simulation: "Roughly speaking, a transition system TS' can simulate transition system TS if every step of TS can be matched by one (or more) steps in TS'. Bisimulation equivalence denotes the possibility of mutual, stepwise simulation." [26, Section 7.1]. A detailed study of bisimulation was conducted by Roggenbach and Majster-Cederbaum [117], and for probabilistic bisimulation by Abate [3]. One application of simulation and bisimulation is to abstract a system to a bisimilar quotient system for verification purposes [26, Section 7.1.1]. While bisimulation transfers CTL [26, Section 7.2], simulation transfers only a fragment of CTL [26, Section 7.5]. Simulation implies trace inclusion, if there are no terminal states [26, Theorem 7.70]. Bisimulation implies trace equivalence [26, Theorem 7.6]. A detailed comparison of bisimulation, simulation and trace equivalence can be found in [26, Section 7.4.2]. These three discrete conformance relations have

one thing in common: They do not distinguish between input and output events. When the input-output distinction is considered, there are additional concerns: input-enabledness/input-receptiveness and quiescence. Input-enabledness holds for a system, if all possible inputs can be applied in every state. For non-input-receptive systems, the system may declare inputs as illegal, based on the state of the system [109]. Quiescence was used by Tretmans for the input-output conformance relation (ioco) [132]. This is basically a synthetic output, which captures the fact that there is no output of the system. Ioco assumes input-enabledness and can be seen as the counterpart of trace conformance for systems with input-output distinction. Two systems are ioco, if the possible outputs of one system are also possible outputs of the other system after the same sequence of events. On the other hand, there are other relations for input-output system, which do not assume input-enabledness, such as refinement calculus [25]. Although most of the conformance relations in this paragraph are built for discrete systems, they can also be applied to infinite transition system. In principal, timed and hybrid system, formulated as infinite transition systems, can use the same conformance relations. However, the meaning of conformance depends heavily on the definition of these transition systems and typically does not capture the specifics of timed and hybrid systems. Therefore, additional conformance relations were proposed, which are based on the discrete conformance relations.

Timed systems extend discrete systems with time and passing time can be considered as an additional real-valued event. The continuous time enables the measurement of time between two events. This has led to timed simulation [130], as well as approximate timed simulation [67], which allows a bounded deviation between the times at which the events happen on both systems. Furthermore, time can be considered as an output, which reduces the need for notions such as quiescence. This was done by *tioco*, a timed version of ioco [81]. A detailed discussion on timed conformance relations can be found in Schmaltz and Tretmans [119].

Hybrid systems are a superclass of both discrete and timed systems. Hence, important points of discrete and timed systems do also arise for hybrid systems. The traces of hybrid systems can be abstracted to timed or discrete sequences and timed or discrete conformance relations can be applied to it. With this approach some information of the systems is lost in the abstraction. In this survey, we focus on hybrid systems, which are input-enabled and where the hybrid part plays an important part, meaning that discrete or timed conformance relations cannot be used for conformance considerations.

As opposed to discrete and timed systems, the states and the output of the physical part of a hybrid system is real-valued. A richer set of traces (real values over continuous time) are possible, compared to discrete sequences of discrete outputs. In particular, such a CPS

model is expected to react to inputs in dense time, producing a dense-time output. A system model that does not define the reaction to a legitimate input at all times will typically be seen as defective. Blocking input events might be possible, but due to the required continuous evolution of the system, these can be assumed to be modelled as implicit self-transitions of a location or transitions to an error location.

Discrete systems can be non-deterministic in the current location and in the set of possible transitions. Hybrid systems can have this discrete non-determinism and additionally non-determinism on the continuous evolution. This can be modelled by replacing differential equations with differential inclusions leading to uncountably many traces of the system, compared to countably many for discrete systems.

For discrete systems, structural coverage (e.g., visiting all states or code coverage) can be used as a measure of testing progress. There are methods to construct a set of test inputs, which are sufficient to prove that the system under test is conformant, as long as the number of finite states can be bounded [38]. Hybrid systems have an uncountably infinite number of states with respect to which coverage must be defined. Therefore, any meaningful coverage metric requires (finite or countable) abstractions of the state space. At this point, discrete coverage measures can be used again. This comes with the additional cost that one has to define an appropriate abstraction. Since the state space is continuous, the computation of a coverage measure may include geometric operations and can be computationally demanding [41]. Metrics of the continuous state space can be used to define a coverage measure, however the choice of metric and its implications are non-trivial.

With the basics of discrete and timed conformance relations and the characteristics of hybrid systems in mind, we now give an overview of hybrid conformance relations.

2.4 Conformance Relations

In this section, we give an overview of existing conformance relations and their test procedures based on the introduction in Section 2.2. We classify conformance relations as presented in Figure 4: *Simulation* [124] is a conformance relation that relates states; *Trace conformance* [41] only relates output traces and not states; *Reachset conformance* [116] abstracts the set of traces and only checks these abstractions; *Approximate simulation* [55] and *approximate trace conformance* [57] are approximate versions of simulation and trace conformance, where the states and output trace respectively only have to be approximately similar. Each class of conformance relations is reviewed and explained in detail in a separate subsection. We structure each subsection into three parts: (i) definition, (ii) properties and transference, and (iii) conformance verification and falsification. After presenting the different conformance

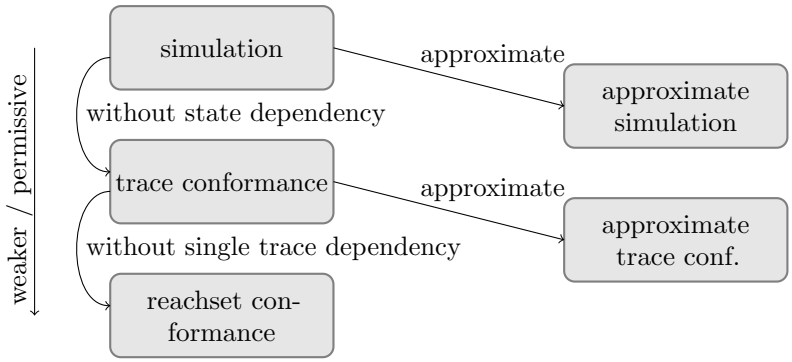


Figure 4: Conformance relations overview.

relations, we compare them in Section 2.5.

While simulation and trace conformance are hybrid adaptions of discrete conformance relations, the other relations do not have an exact discrete counterpart. Generally, hybrid systems could be transformed to transition systems or abstracted to discrete systems on which traditional conformance relations can be applied. For instance, in some cases discrete abstractions of hybrid systems can be used for model checking [19, 124, 34] and control [80, 46, 124]. One abstraction-based conformance relation is *qualitative reasoning input output conformance* (qrioco) [10, 32]. Qualitative reasoning models are used, which abstract from concrete system behavior and states by providing a qualitative description of (*i*) system dynamics based on qualitative differential equations, where only the direction of change of a state is described, and (*ii*) so-called qualitative states, i. e., discrete equivalence classes over the continuous states. Two systems S_I, S_A are qualitative-reasoning-input-output-conformant, if the equivalence classes of the states and the derivatives are the same for the system traces. Brandl et al. [31] propose the critera domain coverage, delta coverage, complete delta coverage, state coverage, and transition coverage on the QR model for test generation. Note that such abstractions are coarse, because only the conformance of discrete abstractions is shown.

2.4.1 Trace conformance relations

The conformance relations in this section can be seen as adaptions for hybrid systems of trace inclusion for discrete systems. Dang [42, 41], as well as van Osch [135, 136], have defined similar conformance notions for hybrid systems, which are inspired by Tretmans input-output conformance (ioco) for discrete transition systems [131].

Definition For system S_I to be trace conformant to system S_A, which we write as $S_I \, \mathtt{conf}_T \, S_A$, the output traces of one system have to be included in the set of traces of

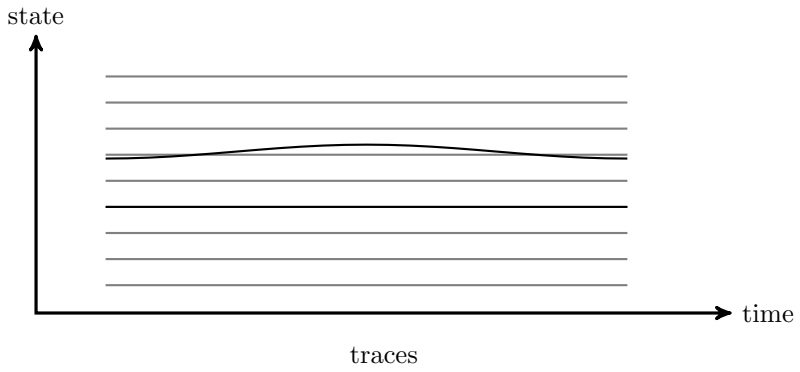


Figure 5: The gray output traces contain only one of the black output traces. The other output trace has a variable slope, the gray ones do not have, and thus the black output traces are not trace conformant to the gray traces.

the other system:

$$\begin{aligned} S_I \,\mathtt{conf}_T\, S_A &\Leftrightarrow \forall u \in \mathit{input}(S_A) : \mathit{otraces}(S_I, u) \subseteq \mathit{otraces}(S_A, u) \\ &\Leftrightarrow \forall u \; \forall T_I \in \mathit{otraces}(S_I, u) \; \exists T_A \in \mathit{otraces}(S_A, u) \; \forall t : T_I(t) = T_A(t). \end{aligned} \tag{1}$$

Following Dang [41], we call this relation *trace conformance*, and an example is visualized in Figure 5. Note that system S_A defines the relevant input set. Therefore, system S_I only has to conform to S_A for the inputs of S_A and there is no restriction for other inputs. This is the reason why the trace conformance relation is only transitive if the input sets are equal.

While Dang defines trace conformance for hybrid automata and on traces, van Osch follows Tretmans more closely and uses hybrid labelled transition systems as the system modelling formalism by defining the relation based on transitions and not based on traces. Van Osch calls the conformance relation *hybrid input-output conformance* (hioco). There exist several other names for basically the same notion of conformance for different formalism. Alur et al. [16] define *language inclusion* for hierarchical hybrid systems, Henzinger et al. [68, 17] *refinements* for hierarchical hybrid systems. Lynch et al. [85] introduce *implementation relation* for hybrid input output automata, whereas Tabuada [124] uses the name *behavioral inclusion*. Another name for trace conformance – inspired from discrete conformance relations – is weak simulation. Grasse [63] presents trajectory propagation and trajectory lifting. Ikeda et al. [73, 39] present the inclusion principle – which is basically trace conformance – for linear systems in the context of decentralized control. Mitsch et al. [92] discuss *projective relational refinement* for refactoring and refining hybrid systems given in differential dynamic logic. Following Quesel [110], we call the systems *trace equivalent* if trace conformance holds in both directions. Alur et al. [19] call it *language equivalence*, whereas Pola et al. [105] use the name *input-output equivalence*.

Properties and transference Since the output traces of S_I are also output traces of S_A if S_I is trace conformant to S_A, properties which hold on all output traces of S_A also hold on all output traces of S_I. For instance, if an ouput is not reachable for system S_A, then it is also not reachable for system S_I. Alur et al. [19] show that trace equivalence transfers linear temporal logic (LTL) properties, but not computational tree logic (CTL) properties. The same should also hold for trace conformant systems, but we found no proof in the literature.

Conformance verification and falsification For deterministic systems and a given input trace conformance can simply be tested by checking if both output traces are equal. For checking the inclusion of a trace of S_I in a non-deterministic system S_A, one has to find the non-deterministic choice for S_A that produces the same output trace. Van Osch [135, 136] approaches this problem by computing a tree where the edges are labeled with inputs or outputs and leaves are labelled with pass or fail. Paths starting at the root of the tree represent traces of the system. Finite time evolutions of the continous parts are included as elements of the discrete test tree. A test execution traverses the tree and finally leads to a pass or fail on the leaf. The main problem with this algorithm is that it is not clear how to generate a memory-limited tree for a given hybrid system (there are infinitely many traces), which checks conformance and covers all behaviours. Structural information from the hybrid systems is not leveraged, but naively transformed into hybrid labeled transition systems and test trees.

Dang [42, 41] proposes a practical approach for checking trace conformance of deterministic systems with a coverage-guided test generation implemented in the test generation tool HTG[2] [41, 43]. Given a hybrid system model S_A which should be checked against a black box implementation S_I, the algorithm generates a trace tree approximately covering the state space of the model S_A. This is done using a discrepancy measure that recognizes regions which are not covered very well. Iteratively, these regions are explored and approximately covered using rapidly exploring random trees. The disparity of the sampled state space is used as a test-end criterion. The result is a test tree which can be used to select the input for the implementation under test and to compare the resulting outputs.

Another approach is used by Mitsch et al. [91] to synthesize a conformance monitor from a model given in differential dynamic logic. It does a sampling based check of trace conformance for measured data against the model.

[2] `https://sites.google.com/site/htgtestgenerationtool/home`

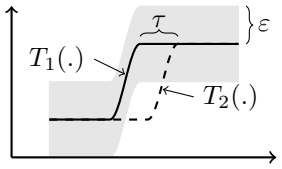


Figure 6: The output trace $T_2(.)$ is similar to output trace $T_1(.)$, but with a time delay τ. The output distance of $T_1(t)$ and $T_2(t)$ is greater than ε for some time t.

2.4.2 Approximate trace conformance relations

A small difference of output traces of two deterministic systems S_I, S_A is sufficient to invalidate trace conformance. However, S_I and S_A can be approximately conformant if their traces remain close to each other. We can use a metric d to quantify the distance of the traces so that ε-approximate trace conformance can be defined as

$$I \,\mathtt{conf}_{\approx}\, S \quad \Leftrightarrow \quad \forall u \in \mathit{inputs}(S)\ \forall T_I \in \mathit{otraces}(I,u)\ \exists T_S \in \mathit{otraces}(S,u) : d(T_I,T_S) \leq \varepsilon.$$

If one uses the metric $d(T_I,T_S) = \sup_t d_o(T_I(t),T_S(t))$ with a metric d_o on the output space, one obtains the *approximate language inclusion* as defined by Girard and Pappas [57]. However, they only used it for comparison to approximate simulation – contrary to Bian and Abate [30] who focused on approximate trace conformance in a probabilistic context. A very special approximate trace conformance relation focusing on heartbeats is presented by Banach et al. [27].

We want to emphasize that transference in the sense of Section 2.2 does not hold for approximate trace conformance relations. Instead, system S_A has to satisfy a robust version $[\mathit{spec}]_{\approx\text{-robust}}$ of the specification to verify *spec* holds on S_I:

$$S_I \,\mathtt{conf}_{\approx}\, S_A \quad \wedge \quad S_A \models [\mathit{spec}]_{\approx\text{-robust}} \qquad \Rightarrow \qquad S_I \models \mathit{spec}$$

Most of the following relations focus on systems with continuous output only, which is motivated by control systems that control a continuous physical quantity and where systems are typically modelled deterministically. An output deviation for deterministic systems can be inadequate for systems whose outputs are time-shifted (delayed) as for the jump-response pattern in Figure 6. The following approximate relations focus on such systems and allow output deviation and time deviation. Especially in the case of non-continuous jumps of the signals, retiming plays an important role. For an easier presentation, we use *retiming functions* [110], which simply scale time (not necessarily continuously).

(τ,ε)-closeness relation Abbas et al. [5] present a conformance relation called (τ,ε)-closeness focusing on deterministic models with continuous outputs. They consider a bounded

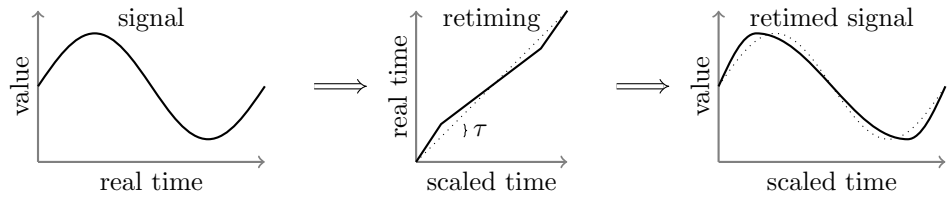


Figure 7: A retiming function transforms a signal.

output deviation ε and additionally a bounded time deviation τ as visualized in Figure 7. The time deviation bound τ can be represented by a retiming function r as

$$\max_t |r(t) - t| \leq \tau.$$

Hence, two output traces T_1 and T_2 are (τ, ε)-close if there exists a τ-bounded retiming $r_{\leq\tau}$ with

$$\forall t : d(T_1(t), T_2(r(t))) \leq \varepsilon \tag{2}$$

and another retiming where (2) also holds, but with interchanged roles of T_1 and T_2. For two points in time t_1 and t_2, the corresponding points in scaled time $r(t_1)$ and $r(t_2)$ potentially have different order: $t_1 < t_2$ but $r(t_1) > r(t_2)$ and thus there could be a local time disorder, which complicates the transference of temporal properties.

Based on the notion of closeness of output traces, Abbas et al. [5] define (τ, ε)-closeness for two deterministic systems S_I and S_A as

$$S_I \, \mathtt{conf}_{(\tau,\varepsilon)-close} \, S_A \quad \Leftrightarrow \quad \forall u \in inputs(S_A) \; \exists r_{\leq\tau} : \sup_t d(T_I(t), T_A(r_{\leq\tau}(t)) \leq \varepsilon,$$

where T_I and T_A are the (single) output traces for the given input and initial state. Note that Abbas et al. [5] define (τ, ε)-closeness for sampled output traces with a finite horizon and hence, their definition is directly applicable to numerically simulated output traces and measured ones. Dependent on the application, a slightly different version of (τ, ε)-closeness defined on hybrid time [5] can be used, which requests that the number of discrete state jumps is the same for both output traces. Mohaqeqi and Mousavi [93] extended (τ, ε)-closeness to also incorporate discrete actions. Additionally, Mohaqeqi et al. [95] study the differences of (τ, ε)-closeness and hioco.

Due to the possible local time disorder, temporal properties can be transfered with some restrictions only. For instance, the property "globally between time 1 and time 3, the velocity is greater than 10" proves only "globally between 1+τ and 3-τ, the velocity is greater than 10-ε" for (τ, ε)-close systems. Abbas et al. [6] prove the transference of such transformed properties in Metric Temporal Logic. The connection of (τ, ε)-closeness to (τ, ε)-approximate simulation is studied by Abbas et al. [4].

To determine the (τ, ε)-closeness between systems, Abbas et al. [4] present an optimization-based approach. They formulate a robustness value that measures the degree of (τ, ε)-closeness. An optimization algorithm, such as simulated annealing, Monte-Carlo techniques [100], or ant colonies optimization [20], can be used to generate inputs minimizing the robustness value. They also present an approach to compute a minimal under-approximation of ε for a given τ and linear switched systems using rapidly exploring random trees (RRTs). These optimization-based approaches are applicable in the sense that their only assumption is that both systems are deterministic and input-enabled. Since such methods usually use sampled traces, Mohaqeqi and Mousavi [94] give error bounds under which sampling of the system does not corrupt the checking of (τ, ε)-closeness against measured data, which is also discussed by Araujo et al. [24]. Aerts et al. [7] present a tool which tests a model versus an implementation on (τ, ε)-closeness. Test-generation is covered with the focus on valid input generation (sound and robust test cases). (τ, ϵ)-closeness [28] was used to check a DC-DC converter.

ε-δ-similarity Quesel [110] introduces ε-δ-similarity, which is (τ, ε)-closeness with $\varepsilon = \delta$ and a preserved time order. Therefore, the retiming r has to satisfy $t_1 < t_2 \Leftrightarrow r(t_1) < r(t_2)$. Several transference theorems are proved for this relation: Region stability, MTL fragment with transference between S_I and S_A in both directions. Quesel does conformance checking using KeYmaera [103] and quantifier elimination. Due to the complexity of quantifier elimination, the approach seems to be limited to simple models.

ε-Skorokhod conformance Deshmukh et al. [45] define the ε-Skorokhod conformance relation, which is identical to ε-δ-similarity with $\delta = \varepsilon$. Since time deviations and output deviations are both bounded by ε, ε-Skorokhod conformance depends heavily on the relative scale of output and time. Contrary to ε-δ-similarity, different conformance bounds can be compared more easily, because it is one dimensional. Dhesmukh et al. prove transference of timed LTL with predicates and freeze quantifier (subsuming STL) with ε-Skorokhod conformance. The authors have implemented a stochastic search-based approach maximizes the Skorokhod distance of the output traces. Since time and output bounds are intertwined, one cannot simply loop over the output traces and compute their distance. Therefore, Deshmukh et al. present a sliding-window-based monitor to check ε-Skorokhod conformance for a given error bound ε. Majumdar and Prabhu [87, 88] present a computational method for quantifying the Skorokhod distance between two hybrid sampled output traces. Skorokhod conformance was used to check an air-fuel ratio controller [45].

2.4.3 (Bi-)simulation relations

The notion of simulation and bisimulation originated from studying discrete structures in theoretical computer science [117]. Tabuada [124] and Frehse [49] give a detailed introduction to simulation relations for hybrid systems. In contrast to trace conformance, simulation relations relate system states rather than outputs. The underlying system formalisms of simulation relations are transition systems.

Definition Continuous and hybrid systems can be formalized as transition systems as done for hybrid (i/o) automata [85, 124] as well as for discrete-time linear systems [128]. A system S_I is simulated by system S_A if there exists a relation R between the state space of S_I and S_A such that for all related states $(s_I, s_A) \in R$ and all state traces x_I of S_I starting at s_I, there exists a state trace x_A of S_A starting at s_A such that

$$\forall t : (x_I(t), x_A(t)) \in R \text{ and } out(x_I(t)) = out(x_A(t)) \tag{3}$$

holds. Note that the classical definition of simulation relation on hybrid automata does not consider inputs. However, by forcing that the evolution x_A and x_I have the same input $u(.)$, this can be incorporated. Cuijpers [40] and Prabhakar et al. [106, 107, 108] define stronger relations called continuous simulation and uniformly continuous (input-output) simulation, which require the relation between states to be (uniformly) continuous.

If the simulation relation holds in both directions simultaneously, the systems are called bisimilar. Bisimulation relations have been used for linear systems [102, 134], for non-linear systems [125], for switching linear systems [105], and for hybrid systems [124, 133, 33]. The theory of bisimulations for dynamical and hybrid systems were unified by Haghverdi et al. [65] using categorical approaches.

Properties and transference One of the benefits of relating states of systems is the transference of CTL properties for bisimilar systems, as shown by Alur et al. [19]. Tabuada et al. [126, 127] show that similarity and bisimilarity of subsystems can be used to construct simulations and bisimulations of the complete system. A discussion of the transference of controllability via simulation relations for linear and nonlinear systems is presented by Ho and Grasse [72, 64]. However, the classical notion of simulation does not transfer stability properties [40]. Stability transference requires a continuous simulation [40], whereas asymptotic stability and input-ouput stability requires uniformly continuous simulation and uniformly continuous input-output simulation, respectively. Rüffer [118] presents a conformance type using comparison systems which similarly transfers stability properties.

Conformance verification and falsification Most computational approaches focus on checking simulation between two models and not between a model and an implementation. For instance, the tool PHAVer [50] can be used to check simulation as detailed by Frehse et al. [52, 49]. This is due to the reason that simulation relates system states, which are typically not possible to be obtained from measurements. For linear systems, there exist methods to compute bisimulation relations between given models using a fixed-point characterization [102, 134]. Tanner and Pappas [129] present necessary und sufficient conditions for a simulation relation between two constrained linear systems. They are able to check simulation relations using a number of linear programming problems. Munteanu and Grasse [96] present a method to compute simulation relations between nonlinear control systems affine in inputs and disturbances. Murthy et al. [98, 99] present the framework BFComp, which computes bisimulation functions using sum-of-squares optimization, δ-decidability over the reals, and counterexample guided search. Yang [139] generalize the scalar-valued simulation functions to vector-valued simulation functions.

2.4.4 Approximate simulation relations

For systems where the continuous dynamics are the main concern, the exact notions of simulation relations can be too restrictive. This led to the generalization of simulation to approximate simulation, used for instance in system biology [97]. The approximate notion does not restrict the outputs of both systems to be the same, but only to remain close enough. By using an ε bound of $\varepsilon = 0$, approximate simulation becomes simulation. The approximate nature is used to related a model or implementation to a simpler model which does not model every detail and thus has some deviation in the outputs. Hence, one has to change properties on transference as discussed for approximate trace conformance relations.

Definition Approximate simulation is defined on metric transition systems which are transition systems combined with a metric d on the output space. The formal definition [55] is as follows: System S_I is ε-approximately simulated by system S_A if there exists a relation R between the state space of S_I and S_A such that for all related states $(s_I, s_A) \in R$ and all state traces $x_I(.)$ of S_I starting at s_I, there exists a state trace $x_A(.)$ of S_A starting at s_A so that

$$\forall t : (x_I(t), x_A(t)) \in R \text{ and } d(out(x_I(t)), out(x_A(t))) \leq \varepsilon \tag{4}$$

holds. One of the main contributors towards approximate simulation has been Girard [54] and co-authors. They defined approximate simulation for continuous systems [57], as well as for hybrid systems [55, 56]. Other researchers built on top of this work and presented approximate

simulations for linear systems in descriptive form [121], for hybrid communicating sequential processes [138], as well as approximate simulation and approximate refinement for metric hybrid input output automata [101]. Tabuada [123] defines (ε, δ)-approximate simulation motivated by stability properties. The relation (ε, δ)-approximate simulation is basically the same as ε-approximate simulation, but with the additional requirement that all states s_I, s_A with $d(out(s_I), out(s_A)) < \delta$ are related by the state relation $(s_I, s_A) \in R$.

The symmetric relation – both systems approximately simulate each other – is called approximate bisimulation. If it holds, the system's output traces are at most ε away of being equivalent. Approximate bisimulations have been defined for constrained linear systems [58, 60], for nonlinear dynamical systems [59], and for hybrid systems [55, 56].

Approximate simulation accepts a deviation on the output measured separately at each point in time. However, approximate simulation is potentially not enough for systems whose outputs are time-shifted (delayed), e.g. a jump-response pattern. For systems where time deviation has to be taken into account, Julius et al. [75] introduce the notion of (ε, δ)-approximate (stochastic) simulation which accepts an output deviation of ε and a time deviation of δ.

Stochastic models and relation are not within the scope of this survey. However, we want to shortly mention that there are also definitions of approximate simulation for linear stochastic systems [76] and hybrid stochastic systems [74, 77]. Abate [3] has conducted a survey on approximate metrics for stochastic processes.

Properties and transference If system S_I is approximately simulated by S_A, for every reachable state of S_I there is a reachable state of S_A with an output difference of the states of at most ε. Unfortunately, we found no theory of transference for approximate simulation besides this reachability transference. However, approximate trace conformance follows from approximate simulation [55, 56] and therefore all transference theorems can be reused. Since approximate simulation is a stronger relation than approximate trace conformance, one could possibly prove even more by combining proofs from simulation transference and approximate trace conformance transference.

To characterize approximate (bi-)simulation of S_I by S_A, Girard et al. introduce simulation functions [55, 57] and bisimulation functions [59, 57], respectively. These functions are inspired by Lyapunov functions which can be used to prove system stability. Bisimulation functions give bounds on the deviation of outputs for a given state pair of both systems and have to be non-increasing. If one can compute a simulation function, this leads to a proof of approximate simulation. The ε-simulation functions generalize simulation functions for (ε, δ)-approximate

simulation [75].

Conformance verification and falsification Several techniques to compute simulation functions have been developed. The first one reformulates the simulation function search as a linear matrix inequality (LMI) problem [58, 60, 77, 86]. The solution of the LMI generates a simulation function. Unfortunately, these methods are restricted to linear systems only. For nonlinear systems, bisimulation functions can be computed by solving a sum-of-squares (SOS) problem [59]. The main advantage of both formulations is that standard solvers for LMI and SOS can be used. However, the scalability of the methods also depends on the numerical robustness and scalability of these tools. The main disadvantage is that these methods do not give a counterexample if they are not able generate a simulation function. Therefore, an interactive approach is not possible and one gains no insight in the system if no simulation function is generated. Yan et al. [138] have proposed a method to compute simulation functions for hybrid communicating sequential processes. If simulation functions of subsystems are available, these can be used to construct a simulation function of the overall system [53]. There are also methods to construct approximate simulating models for a given model [62, 104] and for control purposes [61].

2.4.5 Reachset conformance relation

So far, we have presented conformance relations that relate traces in an exact or approximate fashion. The following relation do something else: It compares abstractions of the set of output traces referred to as $otraces(S, u)$.

Roehm et al. [116] introduce a conformance relation called reachset conformance. It intends to transfer verification results obtained via reachability analysis. Reachability analysis tools compute (an overapproximation of) the set of reachable states – also called reachsets – of a model S for points in time t and inputs u (see Figure 8):

$$Reach_t(S, u) = \{T(t) \mid T \in otraces(S, u)\}.$$

Reachable sets make it possible to verify if a set of forbidden states can be reached. This is especially useful for non-deterministic systems, where small deviations of the system behaviour from the ideal behaviour without uncertainty should also be safe.

Definition Formally, the reachset conformance between S_I and S_A is

$$S_I \,\texttt{conf}_R\, S_A \quad \Leftrightarrow \quad \forall t : \forall u \in inputs : Reach_t(S_I, u) \subseteq Reach_t(S_A, u).$$

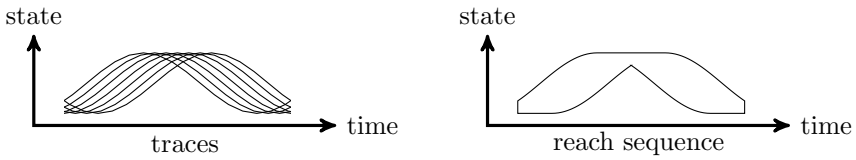


Figure 8: The reach sequence of a set of traces as published by Roehm et al. [116].

Formulated on output traces, the reachset conformance relation can also be defined as

$$\forall u : \forall T_I \in otraces(S_I, u) : \forall t : \exists T_A \in otraces(S_A, u) : T_I(t) = T_A(t). \tag{5}$$

We call the weaker version abstracting time *weak reachset conformance*:

$$S_I \,\texttt{conf}_{\,WR}\, S_A \quad \Leftrightarrow \quad \forall u \in inputs : \bigcup_t Reach_t(S_I, u) \subseteq \bigcup_t Reach_t(S_A, u).$$

Loos et al. [84] introduce differential refinement logic dR$\mathcal{L}$, which syntactically extends differential dynamic logic with a refinement operator on hybrid systems. If the conformance relation is not mixed with the system models, dR$\mathcal{L}$ is identical to weak reachset conformance. However, since mixing of the refinement operator with differential dynamical logic is possible, other conformance relations could also be constructed. Wang et al. [137] define approximate reachability equivalence, where the reachable sets only have to be approximately equal.

Properties and transference Reachset conformance and weak reachset conformance both transfer non-reachability: A state not reachable for S_A is also not reachable for S_I [116]. Although the reachset conformance relation is focused on transfering safety properties from an abstract model (verified with reachability analysis) to an implementation, it also transfers temporal properties that can be formulated in Reachset Temporal Logic [115].

Conformance verification and falsification For two systems S_I and S_A amenable to reachability analysis, reachsets can be computed and checked for reachset conformance. If reachability analysis is not applicable, rapidly exploring random trees (RRT) can be used for black-box models [14], as well as measured data from real systems [116], to underapproximate the reachable sets. Hence, output traces can be seen as sampled elements of the reachsets and used for conformance falsification as visualized in Figure 3. In this case, we have to test the inclusion of the output trace in the reachsets for all points in time which can be done in parallel.

To guide the reachset conformance testing, Roehm et al. [116] introduce a coverage measure based on reachability. It uses the reachsets of the model S_A to select a small input set which approximately covers the reachable space of S_A. The approach can be seen as the

reachability-based version of the coverage measure of Dang [41]. Reachset conformance has already been used to check conformance of a model of walking humans [83].

2.5 Comparison

In the last section, we have reviewed available conformance relations. In this section, we relate the different conformance notions as summarized in Figure 3. We show how to select the right conformance relation with respect to the application context and give some guiding examples to better explain the relations.

2.5.1 Choice of conformance relation

It is not always clear which relation should be used for a given model or for a given scenario. Therefore, we provide some guidance on how to select the right conformance relation. The important properties are summarized in Table 1.

If one wants to formally verify that conformance holds between two systems S_I and S_A, one should use (bi-)simulation or approximate simulation. The reason is that there are no methods to formally prove other conformance relations directly (from simulation follows trace conformance and reachset conformance). Keep in mind that for a formal proof the systems S_I and S_A are required to be white box models (e.g. differential equations have to be known). In general, simulation should be prefered over approximate simulation, when possible, unless both systems deviate from each other to some extent. In that case, approximate simulation can be used to relate the systems. If computational tree logic (CTL) properties are of interest instead of linear-time properties, bisimulation has to be used in place of simulation. Note that formal conformance verification takes a considerable effort and thus is restricted to small system models on relatively high abstraction levels only.

When formal conformance verification cannot be used, one has to resort to conformance testing. For (non-temporal) safety properties, reachset conformance should be used. In this case, while trace conformance or simulation could also be used, they are stricter than is needed to transfer properties of interest, possibly making it more difficult to achieve conformant models. Keep in mind that the published reachset conformance testing method requires the system S_A to be a white box model and amenable to reachability analysis tools, such as Cora [12], SpaceEx [51], and Flow* [35].

If the property to be shown is a more general MTL or TLTL property, trace conformance can be used to relate both systems. In the case when output deviations between both systems are possible, approximate trace conformance is the first choice, because it transfers temporal properties. However, one has to consider approximate conformance relations do not transfer

Table 1: Properties, which guide the conformance relation selection process.

Conformance Relation	System S_I	System S_A	Conformance Checking Focus
Bisimulation	white box	white box	Formal Verification
Simulation	white box	white box	Formal Verification
Approximate Simulation	white box	white box	Formal Verification
Trace Conformance	black box	black box	Testing
Approximate Trace Conformance	black box	black box	Testing
Reachset Conformance	black box	white box	Testing

Conformance Relation	Transference of
Bisimulation	safety, TCTL, TLTL, MTL, STL
Simulation	safety, TCTL (partially), TLTL, MTL, STL
Approximate Simulation	similar to simulation[2]
Trace Conformance	safety, TLTL, MTL, STL
Approximate Trace Conformance	similar to trace conformance[2]
Reachset Conformance	safety, RTL

the exact temporal property, but a slightly changed one [6]. The approach can also be used for black-box models and systems with only their inputs and outputs being accessible.

2.5.2 Simulation vs. Trace Conformance vs. Reachset Conformance

Simulation and trace conformance are equivalent for deterministic systems (with a single initial state), as discussed by many authors [57, 124, 134, 85, 105]. In addition, reachset conformance and trace conformance are also equivalent for deterministic systems [116]. In the case of non-deterministic systems, distinguishing these relations is relevant: simulation implies trace conformance [124] and trace conformance implies reachset conformance [116]. Note that by comparing the definitions of reachset conformance in (5) and trace conformance in (1), the only difference is the different ordering of the quantifiers $\forall t$ and $\exists T_A$.

We illustrate the difference between the relations on three non-deterministic models given in Figure 9 with two continuous state dimensions x and y, the continuous output x, and without inputs.

- The model M1 has two output traces: a sine and a cosine function for initial state $x = 0$ and $y = 1$.
- The model M2 selects at time $\pi/2$ to which function, sine or cosine, it switches after

[2]Property has to be transformed upon transference due to the approximate conformance relation.

time π has passed.

- The model M3 can switch between a sine and a cosine function after time π has passed.

We interpret any discrete state transition as urgent, so that a transition must be taken after every half period represented by the transition label $x = 0$ [90]. The relations between these three models are illustrated in Figure 10. For reachset conformance or trace conformance, one simply has to check the inclusion of the reachsets or the set of output traces in one another. Since M3 is not trace conformant to M1, M3 is not simulated by M1. M1 is simulated by M2 with the state relation R which simply relates identical states: $(s_1, s_2) \in R$, if $s_1 = s_2$. Although M2 and M3 have the same set of output traces, M3 is not simulated by M2. If the required state relation R between M3 and M2 exists, then a state of M2 corresponds to the discrete state *down* with continuous state $x = 0, y = 1$ of model M3. This state of M2 does not exist because the states of M2 can either evolve only up or only down for the following half period and a given state, which ist not the case for M3.

Note that the main source of non-determinism in the example above comes from the discrete transitions. Therefore, the example would also work if the models would be abstracted to discrete models by removing the differential equations. Van der Schaft [134] provides an example about the difference between simulation and trace conformance. Roehm et al. [116] do the same for reachset conformance and trace conformance. In these example, the source of non-determinism is the continuous side and this shows that the differences between the relations do also persist if the source of non-determinism does not come from the discrete side.

2.5.3 Approximate language inclusion vs. (τ, ε)-closeness vs. Skorokhod conformance

The relations *approximate language inclusion*, (τ, ε)*-closeness*, and *Skorokhod conformance* are approximate trace conformance relations and each uses a metric to compute the distance between two output traces. The metric defines which output traces are considered as close. This immediately raises the question of the differences between these relations. To solve this, we discuss the differences by an example. In the following, we consider a deterministic system S, choose one output trace $T(.)$, and characterize the different sets of output traces which are close to $T(.)$. For ease of presentation, we will assume that T is one-dimensional.

Approximate language inclusion requires ε-close output traces to vary at most ε for all times. Therefore, all output traces which are ε-close to $T(.)$ can be represented by tubes which have the same representation as reachset sequences (and are intervals for one-dimensional spaces)

$$R_1(t) := [T(t) - \varepsilon, T(t) + \varepsilon]$$

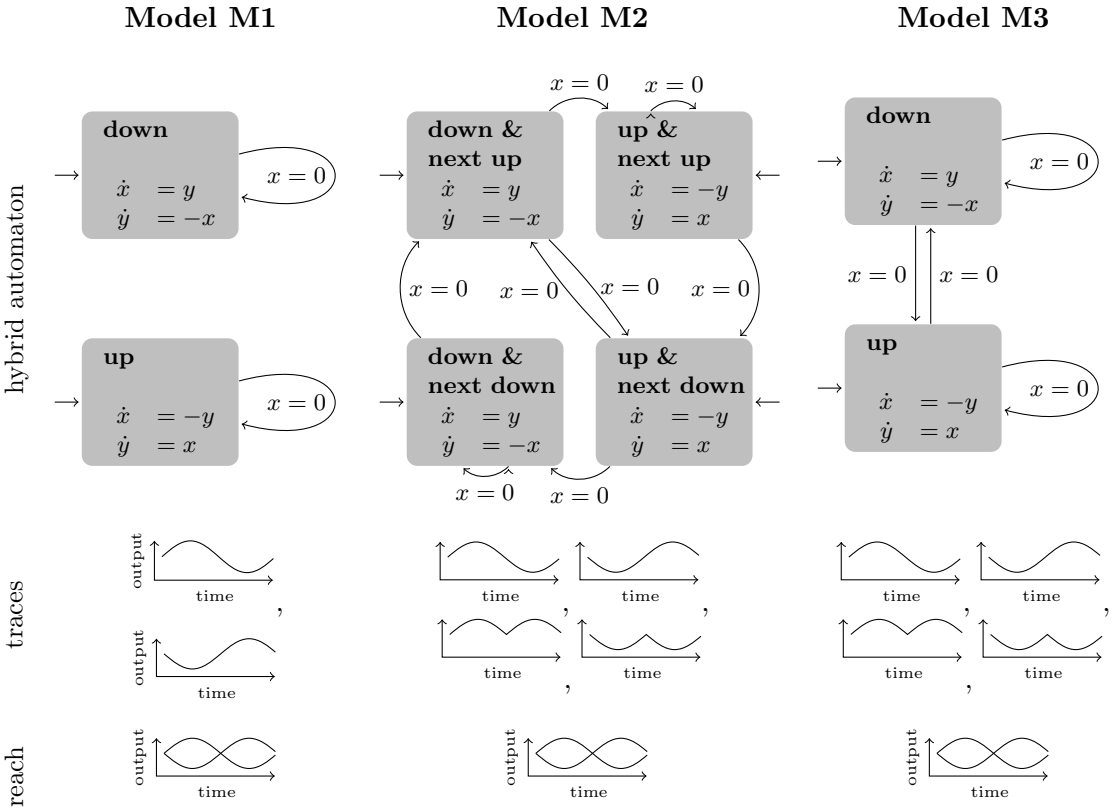


Figure 9: Three example models and their reachsets and output traces.

col reachset conf. to *row*

	M1	M2	M3
M1	yes	yes	yes
M2	yes	yes	yes
M3	yes	yes	yes

col trace conf. to *row*

	M1	M2	M3
M1	yes	no	no
M2	yes	yes	yes
M3	yes	yes	yes

col simulated by *row*

	M1	M2	M3
M1	yes	no	no
M2	yes	yes	no
M3	yes	yes	yes

Figure 10: Comparison of conformance for different models.

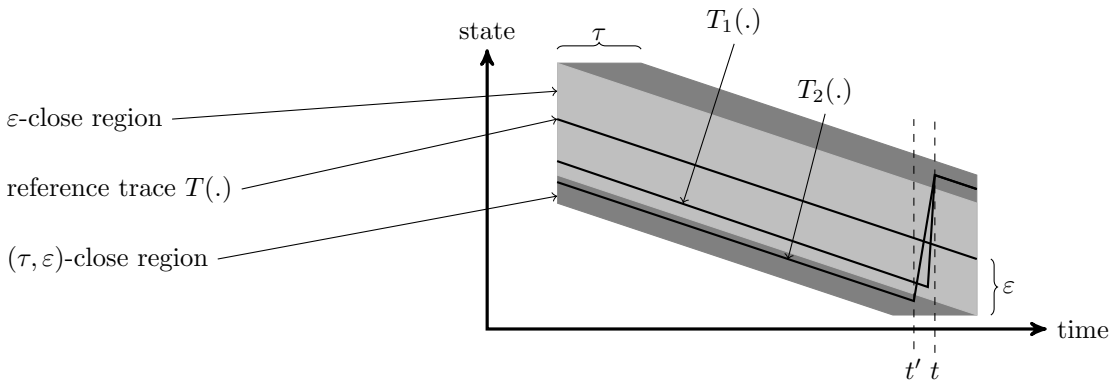


Figure 11: Visualization of reachsets, which represent reference-trace-close traces for approximate language inclusion and (τ, ε)-closeness, as well as two sample traces.

and for an output trace T' the check simplifies to checking $T'(t) \in R_1(t)$ for all times t. Similarly, (τ, ε)-close traces can be represented by

$$R_2(t) := \bigcup_{t' \in [t-\tau, t+\tau]} [T(t') - \varepsilon, T(t') + \varepsilon],$$

which includes $R_1(t)$ and therefore is less restrictive. Conformance relations, where the output traces are ε-close to $T(.)$ can be represented by a reachset, have one important advantage: For every point in time, we only have to check inclusion, which can be done in parallel, because $T'(t_1) \in R_1(t_1)$ and $T'(t_2) \in R_1(t_2)$ are uncorrelated for different times t_1, t_2.

Skorokhod conformance and $\varepsilon-\delta$-similarity are relations without the possibility to represent output traces which are ε-close to $T(.)$ as reach sequences. The main reason, which prevents the representation, is a requirement on the time domain: Different points in time t, t' are required to have the same temporal ordering after retiming. As an example, consider the two output traces T_1, T_2 visualized in Figure 11: Although both output traces share the same value at t, this point in time is the very reason why T_2 – as opposed to T_1 – is not ε-Skorokhod close to the reference trace. The time t has to be retimed to a time $r(t) < t$ for T_2, because the distance between $T_2(t)$ and $T(t)$ is more than ε. However, the same applies to t' and one would need to change the temporal occurrence ($r(t') > r(t)$, but $t' < t$) which is not allowed for these relations. Therefore, we have to consider the whole time domain at once for checking conformance. For $\tau > \varepsilon$ we have that

$$\varepsilon\text{-close traces to } T(.) \subset \varepsilon\text{-Skorokhod close traces to } T(.) \subset (\tau, \varepsilon)\text{-close traces to } T(.).$$

2.5.4 Approximate trace relations vs. approximate simulation

For the approximate versions of trace conformance and simulation, the differences are similar as for the exact versions. Girard et al. [59, 55, 56] prove that approximate trace conformance

follows from approximate simulation. By setting the parameters of approximate relations to zero ($\varepsilon = 0$), one obtains the exact relations. For instance, with $\tau = 0$ and $\varepsilon = 0$, the (τ, ϵ)-conformance is equivalent to hioco [5].

Our short comparison of the conformance relations neglects the fact that the relations are defined on different modelling formalisms in the respective papers. For a mathematically rigorous comparison of two conformance relations $\texttt{conf}_1, \texttt{conf}_2$, one has to use a transformation T to transform $\texttt{conf}_1$ and its modelling formalism to $\texttt{conf}_2$:

$$S_I \,\texttt{conf}_1\, S_A \quad \Leftrightarrow \quad T(S_I) \,\texttt{conf}_2\, T(S_A)$$

for two systems S_I, S_A. Khakpour and Mousavi [79] do this for hybrid input-output conformance, (τ, ε)-conformance, and approximate bisimulation and their underlying formalisms: Hybrid labeled transition systems (HLTS), hybrid-timed state sequence systems (HSS), and metric transition systems (MTS). They define a transformation from HLTS to MTS and prove that hioco is equivalent to the exact simulation relation for input-enabled and deterministic models. Furthermore, they prove that (τ, ε)-conformance is equivalent to approximate simulation relation for their defined transformation from HSS to MTS.

2.6 Input Selection for Conformance Testing

Lee [82] has defined conformance testing as the process of testing a white-box model against a black-box refinement. In this section, we focus on how to select relevant inputs from the large number of possible inputs. We follow Lee and assume that we want to check $S_I \,\texttt{conf}\, S_A$ with a black-box refinement S_I and a white-box abstract system S_A. The goal of input selection for conformance testing is to generate inputs leading to non-conformant behaviour or to give a high confidence that we have tested conformance sufficiently, e.g., using a test-end criterion. Inputs should be selected such that we find non-conformant behaviour fast (falsification) and that we expose relevant behavior (coverage) with a minimal number of tests. It must be said that the topic of input selection for conformance of CPS is still very much under research at this point, and not really as mature as for discrete systems. Therefore, in the remainder we give only a brief overview of existing work (cf. [8]).

Coverage A notion of coverage can support the input selection towards diversification of input selection. For systems with a continuous input and state space, a coverage measure can only be approximative and heuristic. For instance, when sampling the input space, it is not a priori known, which are the interesting regions. Covering the input space is not a good coverage criterion, because a good coverage of the input space may not lead to a good coverage

of the state space. A possible approach is to approximatively cover the reachable state space of S_A with a finite number of tests [48, 47, 78, 22]. This ensures that the inputs are selected such that sufficiently different behaviours of the system are shown. Brandl et al. [31] measure coverage based on the coverage of a discrete abstraction of the continuous state space, called qualitative reasoning models.

If the reachable state space cannot be estimated accurately, we can still try to find inputs that identify novel behavior and increase the diversity of the reached states. Dang [41, 44] uses a measure to quantify diversity and to automatically select a state space area which is not sufficiently explored. This approach converges to a full reachable state space coverage of S_A [41]. Rapidly-exploring random trees are used to find input exploring the selected area and are also used for exploring the reachable set of non-deterministic systems. Roehm et al. [116] introduce a coverage measure which uses reachable sets instead of RRTs to approximate the reachable state space. Another possibility for generating inputs covering relevant behaviour is the use of mutant-based methods [9, 11, 114].

Covering the reachable state space of S_A can be seen as the continuous counterpart of statement coverage, e.g., [21, 23]. It makes the implicit assumption that all paths to a given state are equivalent with respect to testing. Since we always compare two systems, the prefered coverage would be a coverage of the Cartesian product of the state spaces of both systems. However, the refinement is typically black-box or its complexity is too high and thus, a coverage of the Cartesian product of the state spaces of the systems is not possible.

Optimization-based falsification Besides using coverage for input generation, incremental optimization of inputs can be used, as visualized in Figure 12. The difference of the outputs for identical inputs is captured by a heuristic and fed back for selecting the next input. In the literature, this approach has been used for falsifying temporal properties [100, 20] and has been transfered to work for falsifying the conformance relations $(\tau - \varepsilon)$-closeness [5] and ε-Skorokhod conformance [45]. Since both are approximate conformance relations, they already carry an implicit notion of distance between traces. Optimization guides the input selection to maximize the distance between the trace of both systems and thus, to find non-conformant behaviour. If a metric is available, this approach can be used for all other present notions of conformance as well. For reachset conformance for instance, the distance of the reach sequence of S_I to the boundary of the enclosing reach sequence of S_A could be used. This helps to guide falsification towards inputs where the inclusion does not hold.

Besides finding the least conformant behaviour, relevant behaviour should be exposed. This can be achieved by combining coverage-based input selection with optimization-based

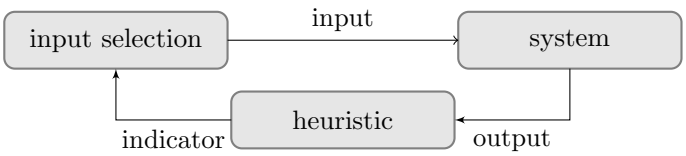


Figure 12: Generic incremental input selection approach.

input selection. The abstract system S_A can be used to approximatively cover its state space and a coverage heuristic can be used to express the confidence that different type of behaviour was exposed.

Since we can only run a finite amount of test cases, we need a test-end criterion at which point we are confident to stop conformance testing. In general, this requires assumptions on the systems under test and hence, this is application specific. Any coverage measure can be used to define a test-end criterion; however, this can lead to test-end criteria, where the test end cannot be interpreted very well or lead to undesired properties of the test-end criterion (cf. [41, Section 1.10]). Dang uses disparity of consecutive tests as a termination criterion [41].

2.7 Conclusion

This survey presents conformance relations for cyber-physical systems. We identify five basic classes of conformance relations: (1) trace conformance relations, (2) approximate trace conformance relations, (3) simulation relations, (4) approximate simulation relations, and (5) reachset conformance relations. We present their definitions, what properties transfer with conformance, and what conformance verification and conformance testing methods exist. We also provide some guidance for the selection of a suitable conformance relation for a given application context, contrasting the characteristics of the different conformance relations from the literature.

While a number of conformance relations have been proposed in the literature, there has not been much research focus on test generation methods for showing these. Open questions include test coverage criteria, guarantees relative to assumptions on system behavior and test input generation algorithms. We believe that, from an industrial standpoint, the applicability of the theoretical conformance relations hinges on providing tailored means to transport these results into practice by systematic testing.

3 Conformance Testing in the Development Process

In this section, we look at conformance testing as used within the UnCoVerCPS approach, and discuss how it relates to the development process from an automotive perspective. In particular, we focus on the notions of trace conformance as described in Section 2.4.1 and reachset conformance as described in Section 2.4.5, and on input generation via optimization-based falsification as described in Section 2.6.

3.1 Non-deterministic Models

A key characteristic of the models used in the UnCoVerCPS context (and hybrid systems verification in general) is that they are typically not deterministic, but contain set-based non-determinism. This means that the simulation of a model is replaced by set-based computations, giving rise to the technique of reachability analysis. This is in contrast to the models that are used in typical development processes today. Classically, models for simulation or control are deterministic, i.e., will produce one single state/output trajectory for a fixed input signal and initial state. When two deterministic models or one model and a set of traces need to be related to each other, this can for instance be done with the help of metrics in the signal space (i.e., the Frechet distance). However, this leads to a notion of similarity (by bounding the measure with some number) that gives no or very conservative guarantees for verification. Hybrid systems verification approaches instead allows one to model of the notion of distance as non-determinism in the model. Whether a signal is conformant to a model is captured by an inclusion relation: all traces which can be produced by a model are conformant to it. This can be used to capture simple pointwise signal distances, as well as more intricate notions of an admissible distance. In other words, since different kinds of uncertainty can be added in different places of a hybrid system model (additive or multiplicative set-based disturbances in differential equations, non-deterministic guards or updates on the transitions, non-determinism between invariants and outgoing guards), we can also model complex notions of conformance directly as an hybrid automaton. In particular, instead of simply adding a disturbance to the observed continuous state, it is possible to define disturbances on inputs and discrete structure of the models. This notion of non-determinism is not common in state-of-the-art modeling in the industry and also has implications for the testing task. Therefore, fitting conformance into the development process largely has to do with integrating the aspect of explicit non-determinism in the models.

3.2 Conformance Checking

The nature of conformance checking problems depends on the conformance relation that is used. In the survey in Section 2, different types relations are discussed in detail. In this section, we will therefore restrict ourselves to trace conformance, which requires that all observed traces are also traces of the non-deterministic model. For the purpose of conformance checking, the non-deterministic model becomes a new type of specification that has to be fulfilled by the real system. This specification can either be tested offline with recorded measurement data or cast into an online monitor that is active at runtime. In both cases, one has to deal with the fact that trace inclusion is undecidable for hybrid automata in general (in fact already for timed automata [66]). However, in our experience, non-deterministic models for physical processes can often be framed in a form where trace inclusion becomes not only decidable, but simple to check. For instance, see the models for the automated driving (AD) use case in Section 5, where a trace conformance check basically amounts to inclusion checks in intervals.

For the online case, state-of-the-art model-based diagnosis functions in industry typically use deterministic models to compute residuals between computed and measured values of physical quantities. These residuals are then compared to a threshold, and if it is violated, there is no conformance, which can for example indicate a defect in some physical component. Online monitoring of non-deterministic models requires online checks for trace inclusion instead, which for our AD use case model is simple, but may be more complex depending on the sources of non-determinism in the model.

From a development process standpoint, offline conformance checking can therefore be seen as a test against a (complex) specification, while online conformance checking amounts to model-based diagnosis with a (complex) model. Both of these activities have direct equivalents in the classical development process.

3.3 Input Generation and Test End Criteria

Since offline conformance checking is essentially testing against a complex specification, this means that classical testing methods can be used for test generation and design of experiments. For example this includes methods for search-based testing/falsification [5], which are already being used in an industrial context. In a nutshell, these methods use optimization algorithms and some measure of specification fulfillment/violation or coverage to generate interesting test cases. Traditionally, the measure is either a handcrafted metric (for example time-to-collision type metrics for automated driving) or based on real-valued semantics of some temporal logic (i.e., STL/MTL [89, 15]). In the case of conformance testing, one only requires a conformance

metric quantifying how robustly a measured trace is included in the model or how far it is outside. In Section 4 we define such a metric, which is then applied to the automated driving use case in Section 5. If the model has more than one discrete state (which is not true in our case), then additionally, test generation methods for state machines, e.g., based on transition coverage can be used.

As far as test end criteria and coverage arguments are concerned, a purely discrete code coverage is not sufficient, but some notion of coverage of the continuous space is needed. In Section 4 we also give a possible approach for that problem. However, the decision when a model can be deemed sufficiently covered by test is one that requires political and societal discussions as well as statistical arguments, resulting in some kind of norm or standard, which is outside the scope of the project.

In summary, other than requiring a quantitative conformance checking procedure yielding a conformance metric, the conformance testing problem can be framed as a spefication-based testing problem, so that infrastructure as well as processes do not require large changes.

3.4 Identification of Conformant Models

The problem of finding a "minimal" conformant model can be cast into the following form: Given a deterministic model (e.g., based on an ordinary differential equations), enlarge model uncertainties enough, so that it is conformant to all measurements. In general, this is a multi-objective optimization problem where the goal is to find Pareto-optimal solutions. For this to work, the model must be given in some parameterized form to find how much uncertainties have to be enlarged.We essentially solve a parameter identification problem of higher dimensionality (i.e., with more parameters) compared to classical system identification, since we additionally require uncertainty bounds. In return, we have a hard yes/no decision on whether a set of parameters is adequate, as opposed to the soft cost functions that are typically used for deterministic models in industrial practice. This set-based parameter identification problem can either be done based on measured data (conformance checking) or together with test generation/design of experiments (conformance testing). In Deliverable 5.2, as well as in Section 4 of this deliverable, we describe such an approach and apply it to the automated driving use case (see Section 5 of this deliverable).

Table 2: UnCoVerCPS based development vs. classical model-based development

UnCoVerCPS based development	**Classical model-based development**
conformance checking	checking bounds on distance measures between traces
offline conformance testing	specification-based testing
online conformance monitoring	model-based diagnosis
test case generation/design of experiments for conformance testing	test case generation/design of experiments for specification based testing
identification of conformant models	parameter identification

3.5 Summary

In summary, from a development process standpoint, conformance testing and the notion of non-deterministic verification models latch on to existing development processes. There is an increase in complexity, because:

- models now have extra parameters modeling the uncertainty,
- computational steps on the model have to be set-based (also including online computations),
- simulation has to be replaced by reachability analysis,

but the advantage is that uncertainties are made explicit in the model and fuzzy notions of similarity are replaced by strict conformance notions. On a high level, the connections to existing process steps are summarized in Table 2.

4 Conformance Testing Methods in UnCoVerCPS

This section describes the conformance testing approaches that have been developed within UnCoVerCPS. First, we present a novel algorithm to identify non-deterministic reachset conformant models from measurement data. While work in this direction has already been performed in deliverable D5.2 [1], the methods presented in D5.2 left room for improvements:

we were only able to identify two conformant Taylor models for two specific maneuvers and the reachable sets of the reachset conformant models were very conservative. Section 4.1 describes the improvements to the methods used in D5.2. Secondly, we give an approach for the automated generation of conformance tests based on Bayesian optimization in Section 4.2. Based on the Gaussian surrogate models generated through Bayesian optimization, we then describe an approach to quantify test coverage. We applied all these approaches to the automated driving use case, which will be described in detail in Section 5.

4.1 New Algorithm to Identify Reachset Conformant Models

This section presents a constructive algorithm that allowed us to (i) find conformant Taylor models for each of the four maneuvers recorded for the DLR vehicle [1] (one Taylor model per maneuver), (ii) to find a generalized Taylor model that is conformant to all recorded measurement data and (iii) to ensure that the models are not too conservative and therefore likely to be of practical use. The application to the automated driving use case will be described in Section 5. Our approach assumes that the models to be identified are ordinary differential equations (ODEs). Even though this approach was developed for the automated driving use case, it is general and can also be applied to different systems. Note that the work presented in the following is currently revisited for resubmission as a journal paper.

Let a (non-conformant) model M be given. This model can for instance be a deterministic model, which approximates the behaviour of a system S. The model can be transformed to a parametric one $M((e_1, \ldots, e_l))$, where the vector $e = (e_1, \ldots, e_l)$ models deviations to the exact model behaviour of M. For instance, on the differential equation $\dot{x} = f(x, u)$ the deviation e_x can be added as an additive term: $\dot{x} = f(x, u) + e_x$. Other deviations, such as multiplicative errors or non-determinism on update functions of hybrid automaton transitions, can be considered as well. As the deviation is not known exactly, we use a set E to represent all possible deviations $e \in E$, and we use the notation $M(E)$ for this non-deterministic model. For the algorithm, we limit ourselves to sets E which are represented by zonotopes.

We start with $E = \{0\}$, which is equivalent to assuming there is no deviation, and increase E until reachset conformance holds, as shown in Algorithm 1. The inputs for Algorithm 1 are the model M with initial set I_M, as well as measurements of the system S for an input $u(.)$. We compute the reachable sets of $M(E)$ and check for reachset conformance against the measurements of S. The check is done with the same reachset conformance testing methods presented in deliverable D5.2 [1]. If we find a counter-example ce, we can compute the direction $d_{ce} \in \mathbb{R}^n$ of the excluded point from the center of the associated reachable set. The direction d_{ce} can be used to increase E by adding a new generator g to the zonotope. The addition of

Algorithm 1 Algorithm to identify non-determism to achieve a reachset conformant model

Input: Abstract model M, initial set I_M, set of measurements $\mathcal{M}$ for input $u(.)$, tuning factor λ

Output: E for which $M(E)$ is reachset conformant

1: Initialize E as zonotope $z(0)$
2: Compute reachable sets $Reach_t(M(E), u(.), I_M)$ for all times t in a given time interval T
3: For all times $t \in T$ search for counter-example ce with $Reach_t(M(E), u(.), I_M)$ and $\mathcal{M}$
4: **if** Counter-example ce exists **then**
5: Compute new normalized generator g with $hgenerator(ce)$
6: Enlarge E by adding $\lambda \cdot g$ as a generator
7: Jump back to 2
8: **else**
9: **return** E
10: **end if**

a generator in the direction of the deviation results in an enlarged zonotope in that direction, which can be used to compute a new zonotope that also includes the counter-example.

The function indicating which generator g to add, depending on the direction d_{ce} of counter-example ce, is called $hgenerator(ce)$ here. The function $hgenerator(ce)$ is implemented as a heuristic. One specific heuristic is as follows. We assume that non-determinism is represented by additive terms $e_i \in E$ for selected differential equation of a deterministic model. Then, the function $hgenerator(ce)$ computes a new generator g based on the share of each dimension in the normalized direction d_{ce} of counter-example ce from the center of the reachable set. This means that we increase the non-determinism of the differential equation of the dimensions which contribute to the non-conformant behaviour.

Additionally, the new generator $g = hgenerator(ce)$ is scaled with a constant λ, which defines the step length of the algorithm. With this parameter, it can be tuned how much the non-determinism is adapted in each iteration. After E is updated, we start again by computing the reachable sets of the new $M(E)$. This iterative process terminates if no counter-example is found anymore, and it is guaranteed to terminate for each measurment. Then, $M(E)$ is reachset conformant with respect to the measurements of S and we return E.

As a result, we are able to automatically increase the non-determinism of an ODE model until it is reachset conformant to a given set of measurements of the system S. In the next section we look at the problem of automated test generation, i.e., the problem of deciding which measurements should be taken to increase the confidence in the conformance of a model to the real system.

4.2 Automated Conformance Test Case Generation

As mentioned earlier in Section 3.3, offline conformance testing is essentially specification-based testing/design of experiments with a specification that is a non-deterministic model. Since in the end we are interested in testing conformance of a model against the real physical system, black box testing methods become necessary as the real system is by definition not a model anymore. Here, search-based testing/falsification approaches [20, 100] have shown great potential in specification-based black box testing which also have been adopted in industry. Regarding conformance testing, search-based testing has also been used in previous work [5, 45] wrt. $(\tau - \epsilon)$-closeness and ϵ-Skorokhod conformance. The work presented in the following differs from related work regarding the conformance relation: we investigate the reachset conformance relation which guarantees the transference of safety properties, and is more permissive wrt. conformance. Furthermore, we propose and evaluate Bayesian optimization as a global optimization technique, driving the test generation for conformance testing.

4.2.1 Search-based Conformance Testing

Search-based testing approaches leverage quantitative metrics on the inputs and outputs of black box systems, which are connected to system specifications, for automatic test case generation by employing optimization algorithms. In contrast to Boolean test results, indicating whether a test case passed or failed, quantitative metrics reflect the degree of passing/failing of the system under test. In case of a passed test, this feedback allows us to estimate how *robustly* a test passed. The robustness value r for test cases allows us to (i) understand the quality of test cases, and (ii) generate or select better test cases, i.e., with a higher probability to detect a fault in the implementation. This is possible because such a quantitative robustness measure allows the use of continuous-valued optimization techniques.

By convention, the robustness is defined such that negative values for robustness imply failed test cases. Thus, given a function $I(p)$ mapping parameters to input signals, global optimization methods can be used for finding input parameters $p \in \mathbb{R}^n$ that result in a negative robustness ($r < 0$). These methods are typically use in conjunction with temporal logic specifications on the system level.

In order to leverage search-based testing for conformance testing wrt. a chosen conformance relation, a quantitative conformance robustness metric r is necessary. Roughly speaking, this means that for a given parametrized input $I(p)$, this metric quantifies how robustly a behavior of the reference system M_R is included in the model M. In order to find the parameters p^*

of an input $I(p^*)$ that falsify conformance between the model M and the reference system M_R, global optimization methods are used in search-based conformance testing to solve the problem of finding the global minimum of robustness

$$p^* = \arg\min_p r(M, I(p), M_R). \tag{6}$$

If this minimum is found positive, the model is proven to be conformant. If any test case is found where $r(M, I(p), M_R) < 0$, this implies that $r(M, I(p^*), M_R) < 0$ and thus the model is not conformant. Since it is not possible in general to show that a global optimum has been found, this necessitates the use of heuristics for determining when to stop testing. An approach for such a heuristic is described in Section 4.3. In the following section we present a conformance robustness metric r that can be used for search-based reachset conformance testing.

4.2.2 Conformance Robustness Metric for Reachset Conformance Testing

As mentioned above, the conformance robustness metric r has to quantify the robustness of conformance between model M and reference system M_R. In previous work within the UnCoVerCPS project we developed different algorithms that were used in reachset conformance checking against measured data in the automated driving use case [116, 1]. However, these algorithms only check if a reference point is included within the reachable set of the model without robustness quantification.

Thus, we need a measure how robustly a reference point is included within a reachable set. Figuratively speaking, reference points close to the borders of the reachable set are less robustly included than reference points near center of a reachable set. Then, given a sequence of reachable sets and a sequence of reference points (i.e., a measured signal) we take the minimum over the robustnesses values over all sequence elements. The resulting number will be positive if the signal is included in the reach sequence and negative otherwise.

To compute the robustness for a single reference point and a reachable set, we first compute the distance to the closest boundary of the reachable set and then normalize the result with respect to the size of the zonotope. Without normalization, small reachable sets would yield much lower robustness values, and dominate the result for sequences, since we take the minimum. In the following we mathematically define the robustness metric.

In the reachability analysis tool CORA, a reachable set R is represented as a sequence of n-dimensional zonotopes Z in generator representation (G-representation):

$$Z = z(c, g_1, \ldots, g_m) := \left\{ c + \sum_{i=1}^{m} \lambda_i g_i \mid \lambda_i \in [-1, 1] \right\}, \tag{7}$$

where $c \in \mathbb{R}^n$ is the center and $g_1, \ldots, g_m \in \mathbb{R}^n$ are the generators of Z. Since zonotope inclusion checks, i.e., checking if a reference point is included in a zonotope, do not scale [116] to higher numbers of generators, we overapproximate the zonotope with a polytope P in halfspace representation consisting of less halfspaces than the zonotope. A n-dimensional polytope P in halfspace representation (H-representation) is defined as

$$P = p(H, k) := \{\ x \in \mathbb{R}^n \mid H \cdot x \leq k\ \}, \tag{8}$$

with $H \in \mathbb{R}^{m \times n}, k \in \mathbb{R}^m$ and is also called a m-polytope. Furthermore, let $H_j \cdot x \leq k_j$ be the representation of the j-th hyperplane of m-polytope P with $j \leq m$, i.e., H_j is j-th row of matrix H. Without loss of generality, we assume that H_j is normalized, i.e., $|H_j| = 1$.

Let c be the center of the zonotope Z that is overapproximated by P. Then the distance d_c^j of c from the hyperplane H_j of P is $d_c^j = H_j \cdot c - k_j$. Similarly, the distance $d_{x_{ref}}$ of a reference point x_{ref} from the j-th hyperplane is $d_{x_{ref}}^j = H_j \cdot x_{ref} - k_j$.

Given a reference point x_{ref} and a m-polytope overapproximation P, we compute the reachset conformance robustness metric r by

$$r = \min_{j \leq m} \frac{d_{x_{ref}}^j}{d_c^j}. \tag{9}$$

Here, the distance of the center of the zonotope to the hyperplane is used to normalize r.

In order to generalize equation (9) to sequences of reference points and reachable sets, we compute $r(I(p))$ with the associated reference trace $x_{ref}(t)$ by

$$r(I(p)) = \min_{t_i} \min_{j \leq m} \frac{d_{x_{ref}}^j(t_i)}{d_c^j(t_i)}. \tag{10}$$

The resulting value of $r(I(p))$ is the associated robustness metric for the input signal $I(p)$.

4.2.3 Bayesian Optimization for Test Generation

An optimization method that is well suited for test generation is Bayesian optimization. The reason for this is the fact that test runs using complex models are usually costly enough to warrant the overhead of the Gaussian regression [112] that is used in this optimization method. This is especially true for reachset conformance testing, since each test requires not only a measurement on the real system or high-fidelity simulation, but also a reachability analysis for the abstract model. Furthermore, the explicit surrogate model of the system that is generated by the Bayesian optimization algorithm can be leveraged for test coverage arguments, as will be described in Section 4.3.

In a nutshell, the idea of using Bayesian optimization for conformance testing is as follows. We assume that a parameterized input trace generator, the reference system (e.g. a high

fidelity simulation model), the abstract model, the reachability algorithm and the conformance metric are all subsumed in a single mathematical function $r(p)$. Given a number of previous test results (i.e., parameter values p_i and the associated robustness values $r(p_i)$), Bayesian optimization builds a Gaussian process regression model of the function r [112]. Since the model is a Gaussian process, it can be used to derive predictions of $r(p)$ for parameter values p that have not been tested yet. More precisely, the predictions $r^{pred}(p)$ are one-dimensional Gaussian curves for each p, i.e., for the probability that $r(p)$ is precisely r we have:

$$P(r|p) = \frac{1}{\sigma_p\sqrt{2\pi}} e^{-1/2((r-\mu_p)/\sigma_p)^2}, \tag{11}$$

where the predicted mean μ_p and the predicted variance σ_p^2 are determined as part of the regression. Note that the σ_p^2 is an epistemic uncertainty, i.e., it represents the "level of knowledge" over that part of the input space and will decrease as test coverage increases. Note that neither μ_p nor σ_p^2 are given in analytical form. Instead, they must be computed for each p separately. Using this prediction different so-called acquisition function can be used to described how desirable a certain p is as a new sample of the optimization loop (in our case, as a next test). Classical acquisition functions are obtained by maximizing expected improvement, probability of improvement, etc., on the Gaussian regression. In our case, also maximizing the probability of non-conformance is a useful option. This underlying second optimization loop is done using another optimizer, which can run much faster, as instead of running the full system under test, only the predictor function for the Gaussian process needs to be called.

A main advantage of using Bayesian optimization is that the notion of exploring as of yet untested areas of the input space is rather naturally combined with the notion of running more tests in the more critical areas of the input space. This is caused by the fact that there can be two reasons for a test to be prioritized by the acquisition functions: (1) the predicted mean robustness is low, and (2) the predicted robustness variance is high, which naturally occurs in untested parts of the state space. Depending on the optimizer setup, these two testing goals can be placed in different trade-offs as needed.

In summary, a Bayesian optimizer for conformance test generation will, starting from a number of initial samples (i.e., boundary cases of the test space): (1) build a Gaussian process regression model given the available samples, (2) optimize an acquisition function over the Gaussian process indicating how interesting a test would be based on the prediction, (3) run this test and add it to the set of samples, and (4) return to (1) unless a test end criterion has been triggered.

4.3 Test Coverage as a Test End Criterion

In general, the question when to stop testing, i.e., when no parameter value p such that $r(I(p)) < 0$ can be found, is difficult to answer. Obviously, testing cannot provide correctness proofs in infinite test spaces, one has to rely on additional assumptions. For example, software testing metrics such as code coverage or MC/DC [36] focus on covering the discrete decision space of a program, with the implicit assumption that each sequence of discrete decisions is sufficiently covered by a single test only. The same is true if transition coverage metrics on automata are used: it is sufficient to trigger each transition once. For hybrid systems in general and conformance testing in particular, this assumption no longer holds, as the continuous dynamics might behave very differently in different parts of the state space. In fact, the models used for the automated driving use case contain no switches at all, so using traditional software testing metrics, they would require just a single test. As this is clearly not sufficient, we will in the following give an approach for coverage in continuous spaces that ties in well with the Bayesian scheme for test generation. Note that this is however just one possibility to define such a metric: other approaches are certainly possible (see Section 2.6).

We propose to use Gaussian process regression not only for test generation, but also as a quantitative approach of defining coverage. For each parameter value p, the value of

$$P(r|p) = \frac{1}{\sigma_p\sqrt{2\pi}} e^{-1/2((r-\mu_p)/\sigma_p)^2} \tag{12}$$

describes the probability that $r(I(p)) = r$ according to the Gaussian process model. The basic idea here is that the integral

$$P(r < 0|p) = \int_{r<0} P(r|p)dr \tag{13}$$

describes the likelihood of still finding a failing test, according to the model. Note that this probability depends both on the predicted mean and the predicted variance at p. Figure 13 visualizes (13) as yellow area. If we now compute the integral

$$\int_{p\in P} P(r < 0|p)dp \tag{14}$$

it has the following interpretation: Given a uniformly distributed random selection of $p \in P$, what is the probability, according to the model that the test will fail? We propose that this is an useful way of quantifying the residual risk that is still left after a number of tests. In particular, for a robustly conformant model, this measure will converge to zero as the number of tests go to infinity, as σ_p^2, will also tend toward zero.

There is however, one caveat with this method: the Gaussian regression for a set of tests is not unique and in fact depends on a number of hyperparameters which (implicitly) encode

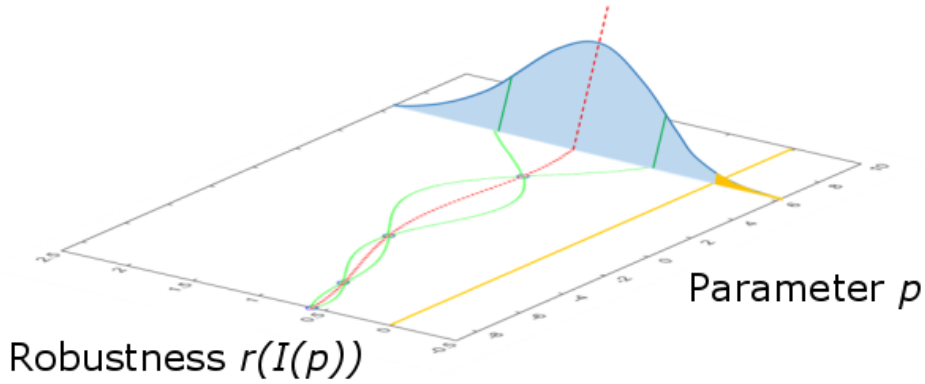


Figure 13: Illustration of the Gaussian process regression for a single parameter p: Blue circles denote actual test cases. The red dashed line is the form of the expected value μ_p along with $\mu_p \pm \sigma_p$ in green. Furthermore, for one selected parameter, the figure illustrates the Gaussian distribution of the probabilities of the potential robustness values for this parameter (see (12)). The yellow area is the area that is captured by (13).

the assumptions on the system under test, and in particular on the generalizability of tests into the surrounding areas of the input space. The Gaussian kernel parameters defining the increase of σ_p^2 with increasing distance from actual tests is of particular importance here. For Bayesian optimization, these hyperparameters are often hidden from the user, and optimized in another loop, yielding the best possible Gaussian process fitting the data (e.g. maximum likelihood estimation). If the Gaussian process is to be used to quantify the quality of a test set, these parameters must instead be visible and explicit. They must be set to reflect the desired coverage on the state space. In fact, for the same set of tests, just by varying the Gaussian kernel parameters, the integral can yield all values in the open interval between 0 and 1. Intuitively speaking, the Gaussian kernel is used is used to quantify with respect to a single test, the probability with which new tests with a certain distance will have a certain deviating value. Practically speaking, the Kernel parameters must be derived from physical assumptions on the system which form the basis of the coverage argument. Possible assumptions may be on Lipschitz constants, sensitivities, bounds on measurement or process noise, etc. However, agreeing on a set of possible parameters is a matter of political and societal agreements and, ultimately, standardization and therefore beyond the scope of this project. However, it should always be noted that all conformance testing coverage metrics obtained from this method are relative to these assumptions and results with different sets of hyperparameters are in general not comparable.

5 Application of Conformance Testing Methods to the Automated Driving Use Case

In this Section, we apply the conformance testing methods from Section 4 to the automated driving (AD) use case. First, in Section 5.1, we describe how conformant models for several maneuvers were identified for the DLR vehicle, using the approach from Section 4.1. Then, in Section 5.2 we show the application of the test generation and coverage quantification methods from Sections 4.2 and 4.3. We selected the automated driving use case to evaluate our new methods as this use case has been explored regarding conformance issues in the most depth and therefore is the best illustration of the application of these techniques.

5.1 Identification of Conformant models for the DLR vehicle

In this section, we give new results on the identification of conformant models for the DLR vehicle, going beyong what was already presented in D5.2. In particular, using the method described in Section 4.1, we were able to achieve models that have tighter reachable sets and are therefore more useful within the UnCoVerCPS framework.

5.1.1 Summary of Measurement Setup for Physical DLR Vehicle

In order to have a self-contained version of this deliverable, we summarize the main information on the experimental setup of how we obtained measurements from the DLR vehicle for conformance testing from D5.2 [1].

The test vehicle (FASCar II) is a Volkswagen Passat TDI from 2009 (Figure 14), which is equipped with a combined differential GPS receiver (DGPS) and inertial navigation system (INS). Four different types of maneuvers with a velocity of $v_x = 10\text{m/s}$ and a maximum lateral acceleration $a_y = 2\text{m/s}^2$ have been recorded at a rate of 100Hz. As visualized in Figure 15, the four maneuvers are

1. **Single lane-change maneuver:** One single lane-change from a right lane to the left lane, which is a typical maneuver for automated vehicles.

2. **Double lane-change maneuver:** After a single lane-change, the vehicle stays on the left lane for a small amount of time and switches back to the initial lane. This is a standard overtaking maneuver.

3. **Fast double lane-change maneuver:** This maneuver is similar to the double-lane change maneuver, but immediately switches back to the right lane, when on the left lane. Such a maneuver occurs when avoiding obstacles on the road and is more dynamic than the double lane-change.

Figure 14: The DLR test vehicle FASCar II.

4. **Slalom maneuver:** To challenge the model with measurements of a more dynamic maneuver, we additionally include a slalom maneuver (see Figure 15(d)).

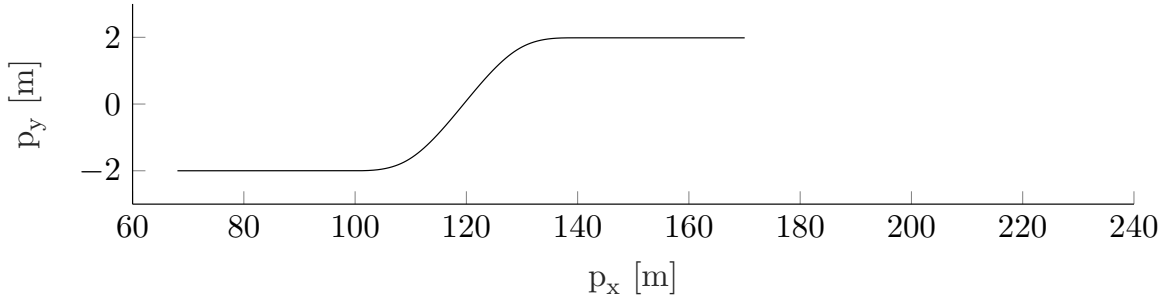
Each maneuver was repeated five times with the average duration of a maneuver being 14.16s. Overall, the total driven distance of the dynamic maneuvers within our measurement data was around 3km. All maneuvers used for conformance testing have been executed in automated driving mode, i. e., closed-loop tracking of a predefined reference trajectory, which is sent from a PC to a closed-loop tracking controller on a dSPACE Autobox.

5.1.2 Taylor Vehicle Model and Non-deterministic Error Structure

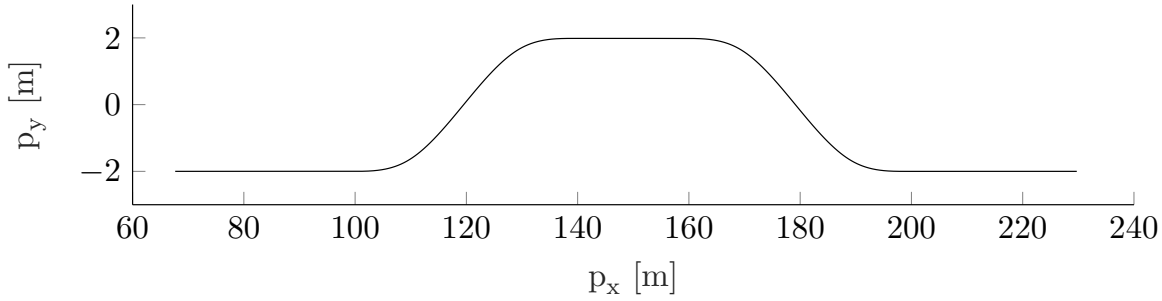
We model the test vehicle described in the section above using a bicycle model as in deliverable D5.2 [1] which is based on the models by Althoff and Dolan [13, 14]. The advantage of the bicycle model is that it sufficiently describes the vehicle dynamics while being simple enough to be amenable to reachability analysis. Since verification might be done online with real time constraints and to speed-up reachability analysis, we further tune the model to speed up the reachability analysis computations.

The state space of our bicycle model is 6-dimensional and has the states $x = (p_x, p_y, \psi, v_x, v_y, \omega)^{\mathrm{T}}$, where p_x, p_y is the position of the vehicle's rear axle center in an earth-fixed coordinate system, ψ is the orientation of the vehicle. The speed of the vehicle's rear axle center is given as $(v_x, v_y)^{\mathrm{T}}$ in the vehicle coordinate system. The velocity components are the respective projections to the vehicle's longitudinal and lateral axis. The vehicle's yaw rate is given as $\dot{\psi} = \omega$. Regarding the differential equations, the interested reader is referred to [1]. Furthermore, the inputs of the model are computed by a non-linear tracking controller which is described in more detail in [70].

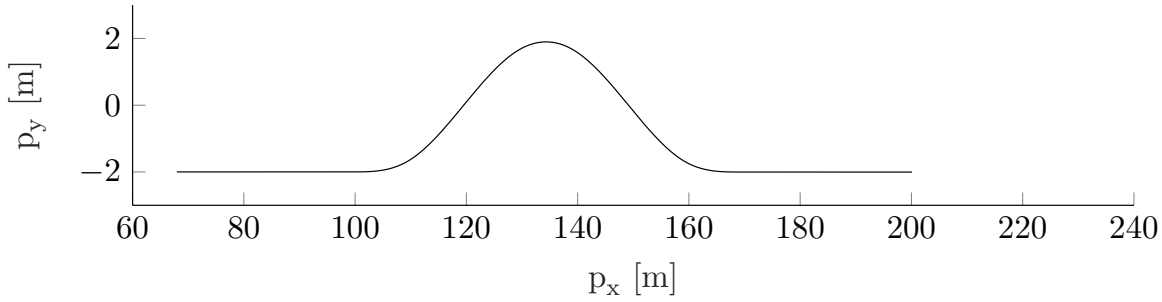
Since we are ultimately interested in the position and orientation of the vehicle, e.g., to detect possible collisions, we use these components as the outputs. As described in Section 4.1, we use the additive error parameters e_x, e_y, and e_ω on the derivatives of the states p_x, p_y and ψ to inject the non-determinism into the model.



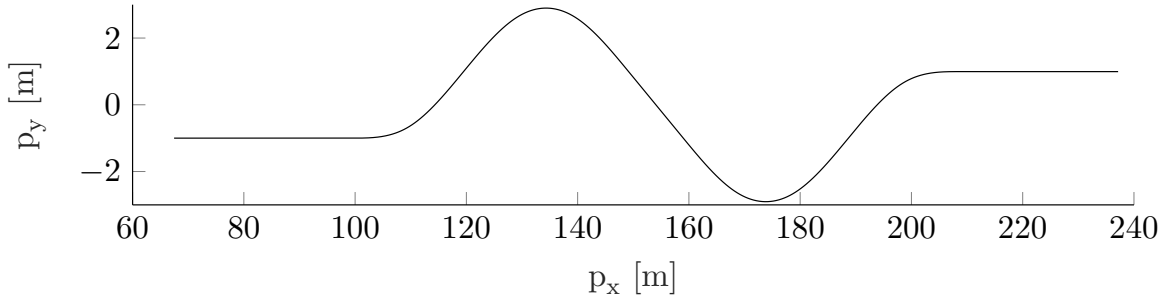
(a) Single lane-change maneuver



(b) Double lane-change maneuver



(c) Fast double lane-change maneuver



(d) Slalom maneuver

Figure 15: The different maneuvers of the experimental setup.

$e_x^m[m]$	$e_y^m[m]$	$e_\psi^m[^\circ]$
0.05	0.05	0.5

Table 3: Measurement errors used in reachset conformance testing

As mentioned above, in order use online reachability and to in order to be able to explore the Pareto fronts of conformant models as mentioned in Section 3.4, computing reachable sets has to be completed within reasonable time. This requires fast computations, potentially within real-time constraints. Therefore, we build a Taylor model of the closed-loop bicycle model which results in a simplification of the Lagrange remainder computations of the reachability analysis. Building the Taylor models has already been explained in Deliverable D5.2 [1] and is therefore omitted in the following. By using Taylor models of the order 2, we achieve a performance gain in reachability analysis up to the factor of 1500.

Note that polynomial approximations of (Taylor) models bears a risk that some behavior is contained in the original model but not in the approximations of the model. However, the original bicycle model is also only a model, which is not the real vehicle. Using a Taylor model does not break any formal arguments, because we are using them for both verification and conformance. The main goal is to have a reachset conformant model and thus, we only have to test that the final model, we are using, is reachset conformant to the measurements of the real vehicle. In the end, we get a reachset conformant Taylor model and not a conformant bicycle model. Note that our approach can also be combined with other advanced online verification approach, for instance with motion primitives [69].

5.1.3 Experimental Results

With the recorded experimental data and the deterministic model of the vehicle, we have all ingredients to build a reachset conformant model with Algorithm 1 from Section 4.1.

For each maneuver we have multiple measured experimental runs. We use the initial points of all runs of the maneuver to build the initial set for the model. Since the measurements contain some sensor errors, we use the bounding box of the initial points enlarged by the sensor errors, presented in Table 3, as initial set I_M of the model.

Regarding the inclusion of measurement points within reachable sets, we check the inclusion of measured data in the reachable sets of the 3-dimensional output space $(p_x, p_y, \psi)^{\mathrm{T}}$. We use the pairwise inclusion check as described in [116]. Regarding measurement errors, we assume that each measured data point of the vehicle position and orientation may be inaccurate up to the values depicted in Table 3.

First, we compute the required non-determinism E^i for each maneuver i independently

with Algorithm 1. The union of the sets E^i, which characterize the non-determinism of each maneuver individually, leads to a model which is reachset conformant for all maneuvers and is called *aggregated model*. Second, a model for all maneuvers is computed by using all maneuvers combined in the Algorithm 1 and we call this model the *collective model*. In the following, we discuss the results and compare the two approaches. Table 4 shows the non-determinism needed such that: (i) all measurements of a single maneuver are reachset conformant to the model and (ii) all measurements of all maneuvers are reachset conformant to the aggregated model and the collective model, respectively.

Our first observation is that we have found reachset conformant models for all maneuvers. All four maneuvers need approximately the same amount of non-determinism for e_x. Compared to the other maneuvers, the slalom model and the fast double lane-change model have more non-determinism for e_y, which seems to be reasonable as these two are the most dynamic maneuvers. For some of the maneuvers, the measurement error on ψ is approximately the same size as the absolute deviation of the measurements from the deterministic model. Since we consider the measurement error, the whole deviation is explainable in the two maneuvers with the non-determinism of the initial set. Furthermore, this case shows the relationship between the error dimensions e_y and e_ψ: Both errors lead to an increased interval width of p_y. The fast double lane-change and the slalom maneuvers have non-determinism for both e_y and e_ψ, to explain the lateral deviation. The double lane-change has only e_ψ and therefore, this non-determinism has to be bigger to account for the whole deviation in p_y.

The aggregated model combines these different non-determinism and thus is a robust but conservative overapproximation. For the collective model, the non-determinism is computed for all maneuvers together in one single run of Algorithm 1. As a result, the non-determinism is slightly lower while still being conformant. Although the aggregated model has a bigger E as the collective model, the amount of non-determinism has the same order of magnitude as the non-determinism for the single maneuvers. This shows that the models are reasonable and the results from single maneuvers generalize. Figure 16 shows the measurements for the single lane-change maneuver. Additionally, the reachset of the conformant model is visualized. Note that the controller has a small damped oscillation after changing the lane. With reachset conformance, this can be easily handled compared to trace conformance, where all the derivatives of the oscillation would have to be considered.

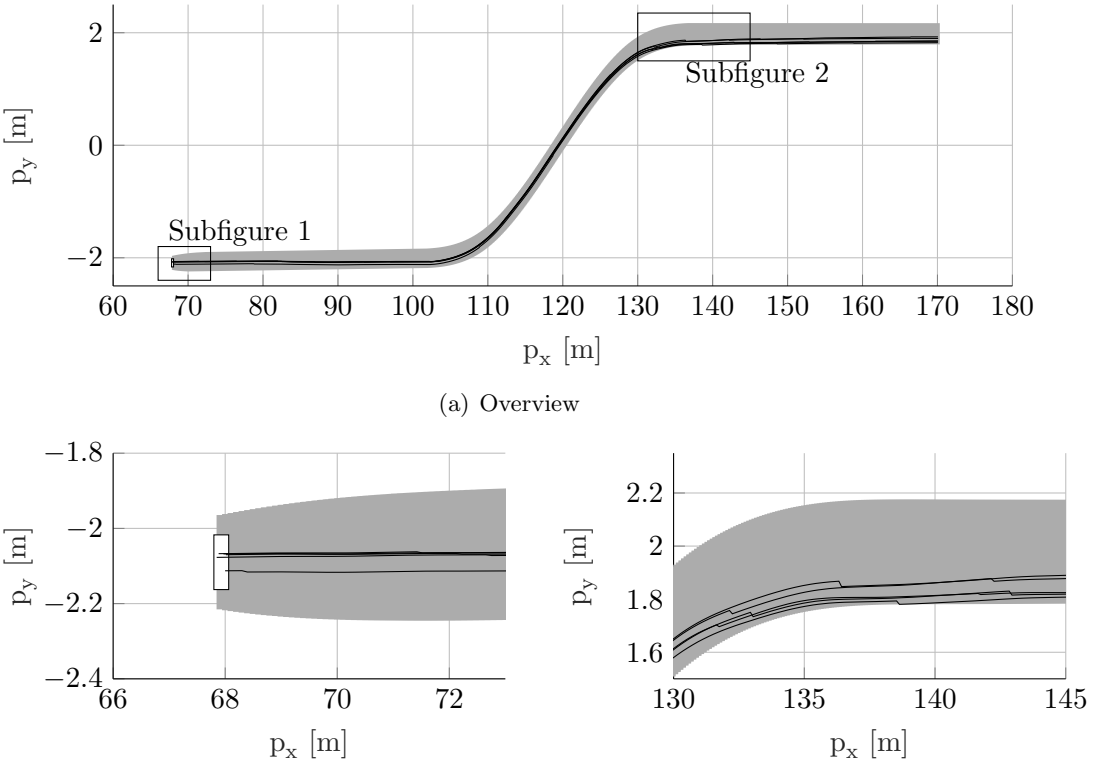
Table 4: Identified intervals for e_* of non-determinism E for the different reachset conformant models.

Model	e_x in m/s	e_y in m/s	e_ψ in rad/s
Double lane-change model	$[-0.1064, 0.1064]$	$[-0.0004, 0.0004]$	$[-0.0140, 0.0140]$
Fast double lane-change model	$[-0.0990, 0.0990]$	$[-0.0129, 0.0129]$	$[-0.0127, 0.0127]$
Single lane-change model	$[-0.1151, 0.1151]$	$[0, 0]$	$[-0.0117, 0.0117]$
Slalom model	$[-0.1253, 0.1253]$	$[-0.0113, 0.0113]$	$[-0.0047, 0.0047]$
Aggregated model	$[-0.1252, 0.1252]$	$[-0.0129, 0.0129]$	$[-0.0140, 0.0140]$
Collective model	$[-0.1136, 0.1136]$	$[-0.0129, 0.0129]$	$[-0.0127, 0.0127]$

To evaluate the applicability of our reachset conformant models for verification purposes, we compute the reachable sets and use their sizes as a benchmark criterion. For real driving scenarios we may only have a small uncertainty in the lateral position p_y. Big uncertainty, such as over 0.5 meter, may lead to situations where we have to assume that our vehicle could be already on an adjacent lane and a collision may be possible. The longitudinal position p_x is less relevant, because one has to consider traffic rules and thus usually has a higher distance to leading and following vehicles [113] than to adjacent lanes.

Table 5 shows the maximal interval widths of the dimensions of the reachable sets for the models, which are reachset conformant for a single maneuver only, as well as for the aggregated model and for the collective model. First of all, the results do not differ very much between the maneuvers. All the models have an interval width in the longitudinal position of at most 0.42m and in the lateral position of at most 0.36m. What we can see is that the range for p_y is significantly increased when the aggregated model is considered, compared to the collective model and even more to the single maneuver models. This reflects the previous discussed fact that the identification of the non-determinism was done independently for each maneuver, which can be an approach to build a more robust model with the disadvantage of increased reachable sets. To get a better feel for the maximal interval widths, we use the measurements to compute a lower bound on the maximal interval width. The values are given in Table 6 and show that a maximal interval width of 0.12m for p_y is needed to enclose all measurements. Our vehicle model and overapproximative approach is responsible for the additional 0.2m maximal interval width. This value could be possibly lowered with a better vehicle model. Note that we only compared the maximal widths here. The interval widths of the reachable sets dependent on the current segment of the driving maneuver and are smaller for some parts of the maneuvers.

One main concern when generating a conformant verification model is if this model generalizes and is conformant in unseen situations. Therefore, we performed a small cross-validation study by generating the reachset conformant model with 3 maneuvers and tested



(a) Overview

(b) Subfigure 1

(c) Subfigure 2

Figure 16: Projection of the measurements and the reachable sets of the conformant model to the position coordinates for the single lane-change maneuver.

Model	Maneuver	p_x in m	p_y in m	ψ in rad
Single man. model	Double lane-change	0.27	0.23	0.02
Single man. model	Fast double lane-change	0.37	0.29	0.02
Single man. model	Single lane-change	0.29	0.20	0.01
Single man. model	Slalom	0.30	0.16	0.01
Aggregated model	Double lane-change	0.33	0.34	0.02
Aggregated model	Fast double lane-change	0.42	0.35	0.02
Aggregated model	Single lane-change	0.34	0.36	0.02
Aggregated model	Slalom	0.33	0.32	0.02
Collective model	Double lane-change	0.29	0.30	0.02
Collective model	Fast double lane-change	0.40	0.31	0.02
Collective model	Single lane-change	0.30	0.31	0.02
Collective model	Slalom	0.29	0.29	0.02

Table 5: Maximal interval widths of the reachable sets for different reachset conformant models and different maneuvers.

Maneuver	p_x^{min} in m	p_y^{min} in m	ψ^{min} in rad
Double lane-change	0.12	0.09	0.006
Fast Double lane-change	0.10	0.12	0.007
Single lane-change	0.16	0.11	0.006
Slalom	0.09	0.11	0.005

Table 6: Maximal difference of all measurements at one point in time. This is a lower bound on the values in Table 5.

the conformance of this model on the last remaining maneuver. The result is that the model is reachset conformant when the double lane-change or the single lane-change maneuver is used as a test maneuver. However, this is not the case for the other two maneuvers. This matches the previous results that these two maneuvers require more non-determinism to explain the measurements. As recording experimental data is expensive, we could only use a limited amount of data for our investigation. In the end, more experimental data is required to build a robust and conformant model, which has to be challenged with data not used for generating the model.

5.1.4 Conclusions

In this section we presented a new algorithm to identify non-deterministic reachset conformant models. To ensure reachset conformance, we bloat deterministic models with non-deterministic parameters. Advantages of the methods are that they are simple to implement and the tradeoff between accuracy and the computational effort can be arbitrarily selected.

In our experimental evaluations, we applied the present algorithm to the DLR automated vehicle, for which we recorded measurements from real experiments. We generated non-deterministic models, to which the real vehicle is reachset conformant to. Since the resulting reachable sets of the non-deterministic model have a diameter of at most 0.31m in lateral direction and 0.40m in longitudinal direction, the model is small enough and ready to be used for verification purposes. With the reachset conformance framework, we are able to foster the application of formal methods for verification of automated vehicles and other applications.

5.2 Search-based Conformance Testing for AD use case

In the following section we present our search-based testing setup for reachset conformance testing an automated driving (AD) use case. Furthermore, we show the applicability of our approach by numerical experiments.

5.2.1 Overview

Figure 17 provides on overview of our search-based reachset conformance testing approach for an AD use case. We use an abstract bicycle model M which is reachset conformance tested against a high-fidelity vehicle model M_R. As our proposed testing approach is currently under development and thus required fast feedback loops, we decided to test the abstract model against a high fidelity model instead of a real physical vehicle. This approach however does not restrict the application of our method to a real vehicle.

The closed-loop high fidelity model consists of 26 physical states and includes multi-body dynamics of a vehicle. In contrast, the closed-loop abstract vehicle model only contains 6 physical states. Both models use the same tracking controller that tries to follow a given reference trajectory. A detailed description on the used models can be found in [14].

Given a sample of a parametrized reference trajectory as input for the tracking controller, we perform a reachability analysis on the abstract model M and a simulation on the high fidelity model. With both outputs, the reachset conformance robustness r is then computed according to Section 4.2.2.

If the model is not conformant, we found an input that falsifies the assumption that the abstract model is reachset conformant to the high fidelity model. If desired, this counter example may then be used to refine the model using the methods described in Section 4.1.

If an abstract model is conformant wrt. to an input, the current value of the conformance metric r is passed to the global optimization algorithm, in our case Bayesian optimization (see Section 4.2.3). The optimization algorithm then proposes a new set of parameters to be tested in the next optimization iteration.

5.2.2 Experimental Setup

As mentioned earlier, both the abstract and the high fidelity model contain the same tracking controller in order to follow reference trajectories that serve as inputs to both models. In theory, the abstract model could also use an abstraction of the tracking controller to reduce complexity for the reachability analysis, however, this would make the conformance test more challenging. In this case, the tracking controller was simple enough so that this was not necessary. We parametrized the reference trajectories of three different standard maneuvers: single-lane change, double-lane change and swerve maneuver (fast double-lane change) as already shown in Figure 15(a)-15(c).

The symmetric single-lane change is parametrized with the three parameters: the desired constant velocity v_0 during the lane change, the width of the lane change w and the total time

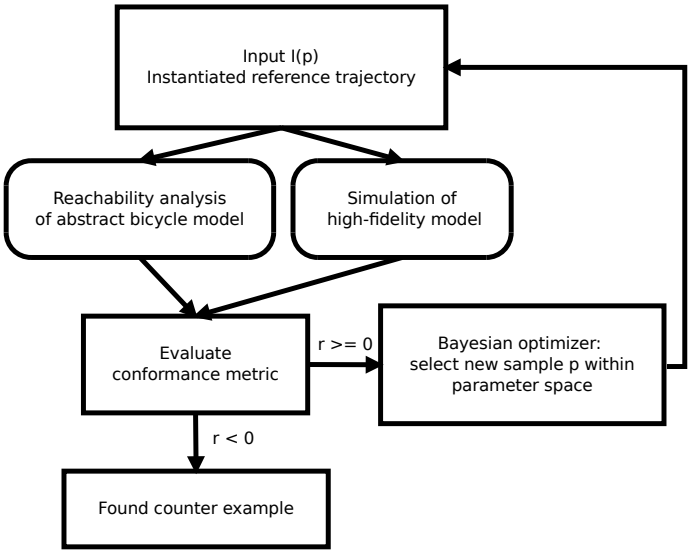


Figure 17: Overview of our search-based reachset conformance testing approach

t the single lane should take. Figuratively, the lane change should be completed after time t where the vertical change in the direction of travel changes continuously until w is reached. We decided on choosing the total time t of the single-lane change as parameter instead of, e.g., the length l of a lane change, since this allowed us to avoid to extreme maneuvers by setting bounds on time t based on reference tools, e.g., Vires Virtual Test Drive (VTD) [2] (see Table 7). Furthermore, the bounds for the width w of the single-lane change are based on the common width of roads in Germany.

Similar to the single-lane change, the symmetric double-lane change is parametrized by the desired velocity v_0 and the width w that is reached before changing back to original lane. Again, to avoid extreme maneuvers, the time t parametrizes the time to perform both single-lane changes within the total double-lane change. Additionally, a fourth parameters t_s is introduced which defines the time the vehicle drives straight at width w before performing the second lane change within the double-lane change maneuver.

The symmetric swerve maneuver has the same three parameters as the single-lane change maneuver. However, the two parameters time t, width w characterize both the lane change parts of the maneuver. As the swerve maneuver is more extreme than the other two maneuvers, the bounds on the parameters are chosen differently (see Table 7). Furthermore, in contrast to the double-lane change, there is no parameter t_s as the vehicle has to change lanes immediately once width w is reached.

For the reachability analysis of the abstract vehicle model, the initial set R_0 was set equal to the initial state x_0^{hifi} of the high-fidelity vehicle model ($R_0 = x_0^{hifi}$). Regarding the

Maneuver	Number of parameters	Parameter bounds
single-lane change	3	$10m/s \leq v_o \leq 20m/s$ $3m \leq w \leq 4m$ $4s \leq t \leq 12s$
double-lane change	4	$10m/s \leq v_o \leq 20m/s$ $3m \leq w \leq 4m$ $4s \leq t \leq 12s$ $2s \leq t_s \leq 5s$
swerve	3	$10m/s \leq v_o \leq 20m/s$ $1.5m \leq w \leq 3m$ $3s \leq t \leq 5s$

Table 7: Overview of the parametrization of the input reference trajectories $I(p)$ with the bound on the parameters of each maneuver

parameters of the conformance metric (see Section 4.2.2), with a focus on passive collision avoidance, reachset conformance was tested wrt. the two states defining the position of the vehicle and the orientation of the vehicle. To compute the robustness of the inclusion of a reference point, the projected 3-dimensional zonotope was approximated by a 1200-polytope with 1200 evenly distributed normal vectors using the same approach as in [116].

In order to have obtain a candidate abstract vehicle model for conformance testing we proceededed as follows. We used the identification Algorithm 1 from Section 4.1 to build a conformant abstract vehicle model. As the measurements that form the basis of the algorithm, we used simulations of the corner cases wrt. the parameter bounds from Table 7. Note that regarding the general problem of specification-based testing, testing corner cases is a common practice within industry. As an example, the parametrized single-lane change maneuver has $2^3 = 8$ corner cases, with

$$(v_0, w, t) \in \{\ (10, 3, 4), (20, 3, 4), (10, 4, 4), (20, 4, 4), \dots\ \}.$$

The swerve maneuver has 8 corner cases as well, whereas the double-lane change has $2^4 = 16$ corner case combinations (since the maneuver is parametrized by 4 parameters). In total, 32 maneuvers where use to build an "assumed to be" conformant abstract vehicle model with Algorithm 1.

The conformant abstract vehicle models was then conformance tested using two different approaches: random testing and search-based reachset conformance testing using a Bayesian optimizer as described in Section 4.2 for each maneuver. For the random testing, we selected 500 random samples of each parameter space of the three maneuvers. Similar, for each maneuver, the search-based testing approach had 500 iterations to find a parametrization of a maneuver that falsifies the assumed to be conformant abstract vehicle model. The

Maneuver	Corner case parameter assignment p	Conformance robustness $r(p)$
single-lane change	$(v_0, w, t) = (20m/s, 3m, 12s)$	$7.02 \cdot 10^{-5}$
double-lane change	$(v_0, w, t, t_s) = (20m/s, 3m, 12s, 5s)$	$5.30 \cdot 10^{-5}$
swerve	$(v_0, w, t) = (20m/s, 3m, 3s)$	$2.74 \cdot 10^{-5}$

Table 8: Overview of the parameter assignments with the minimal robustness within the corner cases of the parameter space for the conformant abstract vehicle model.

Bayesian optimizer was initialized with robustness values of all the corner case parameter space assignments of each maneuver, to form the basis of the first Gaussian regression. Table 8 shows the parameter assignments out of the set of corner cases of each maneuver that returned the lowest conformance robustness for the conformant abstract vehicle model. Note that, since the conformant abstract vehicle model was identified to be conformant to all corner cases of the three maneuvers, all robustnesses for the corner case parameters are positive. As an interesting fact, for the single-lane change maneuver, the parameter assignment with $(v_0, w, t) = (20m/s, 3m, 12s)$ return the lowest robustness. Normally, one would assume that this would be the case for the single lane change with the shorter duration $t = 4s$. A similar observation can be made for the double-lane change, additionally, here the robustness with $t_s = 5s$ is lower than with $t_s = 2s$. In contrast, for the swerve maneuver, human intuition would be right since the lowest robustness is returned for the most extreme choice of parameters $(v_0, w, t) = (20m/s, 3m, 3s)$.

5.2.3 Experimental Results

Table 9 shows for each maneuver the parameter assignments which tested with the lowest conformance robustness out of the 500 random test cases. We observe that none of the test cases return a robustness $r < 0$, thus, no counter example for reachset conformance was found. Furthermore, no test case returned a lower robustness than corner cases from Table 8. Similar, Table 10 shows for each maneuver the parameter assignments that returned the lowest robustness within the search-based conformance testing approach. While also no falsifying test case was found, the search-based approach using the Bayesian optimizer found test cases that returned a lower robustness than the 500 random test cases. Yielding test cases with lower robustnesses in the search-based testing approach results from the Bayesian optimizer's surrogate model of the robustness landscape. Here, the surrogate model allows for a more thorough search in parameter regions that are likely to return low robustnesses. As an example, Figure 18 and Figure 19 show scatter plots of the evaluated parameter assignments in the respective testing approaches. For the search-based approach, we observe that considerably more test cases were chosen with high velocities $v_0 \approx 20m/s$ and widths $w \in [2.5m, 3m]$.

Thus, the test case assignment of $(v_0, w, t) = (19.99m/s, 2.54m, 3.01s)$ was found with a lower robustness than the random test cases for the swerve maneuver.

Figures 20 to 26 show Voronoi plots of the robustness landscape of the test cases computed by the search-based testing approach. The Voronoi plots show the projection of two parameters of the parameter space of a maneuver on the x- and y-axis with the respective robustness value of a test case on the z-axis, taking the minimum of the robustness values, if two tests have the same x and y-values. Furthermore, these plots interpolate unselected test cases with the values of actual test cases that are closest w.r.t. the Euclidean distance. In other words, each plateau in the plot cooresponds to one test in its center and all z-values visible in the plots have actually been observed in tests.

One major observation is that in all Vornoi plots we observe monotonicity towards high values of velocity v_0 for all maneuvers. As we explained before, the corner cases of the parameter space of each maneuver were used to build a conformant model in the first place. Some of the corner test cases yield the lowest observed robustness of all test cases and thus are the lowest robustness points in the almost monotonic robustness landscape. We conjecture that for some use cases, the problem of identifying robust conformant models might be reducible to identifying a model based on the corner cases of the parameter space. While the UnCoVerCPS approach already reduces the complex and expensive process of system testing to testing conformance of models, the effort of finding robust conformant models in the first place could be reduced considerably.

This observation also highlights one of the advantages of using the UnCoVerCPS method as opposed to standard black box testing approaches as they are used in the industry. Since the formal approach to controller design reduces the testing task to conformance testing only physical models and the (often relatively simple) closed-loop component of a tracking controller, the testing task is a lot simpler. Physical models often fulfill continuity or monotonicity assumptions that reduce the number of required tests significantly as opposed to black-box tests also including the planner software. Since UnCoVerCPS uses a formal approach for planner design, significant portions of the testing effort can be reduced by relying on formal proofs.

Maneuver	Parameter assignment p	Robustness $r(p)$
single-lane change	$(v_0, w, t) = (19.02m/s, 3.18m, 11.59s)$	$7.78 \cdot 10^{-5}$
double-lane change	$(v_0, w, t, t_s) = (19.87m/s, 3.12m, 11.07s, 4.73s)$	$6.28 \cdot 10^{-5}$
swerve	$(v_0, w, t) = (19.94m/s, 2.70m, 3.03s)$	$3.78 \cdot 10^{-5}$

Table 9: Overview of the parameter assignments with the minimal robustness within the 500 random test for each maneuver.

Maneuver	Parameter assignment p	Robustness $r(p)$
single-lane change	$(v_0, w, t) = (19.99m/s, 3.18m, 11.97s)$	$7.09 \cdot 10^{-5}$
double-lane change	$(v_0, w, t, t_s) = (19.96m/s, 3.03m, 11.89s, 4.99s)$	$5.38 \cdot 10^{-5}$
swerve	$(v_0, w, t) = (19.99m/s, 2.54m, 3.01s)$	$3.29 \cdot 10^{-5}$

Table 10: Overview of the parameter assignments with the minimal robustness found with the search-based reachset conformance testing approach (using a Bayesian optimizer with 500 iterations).

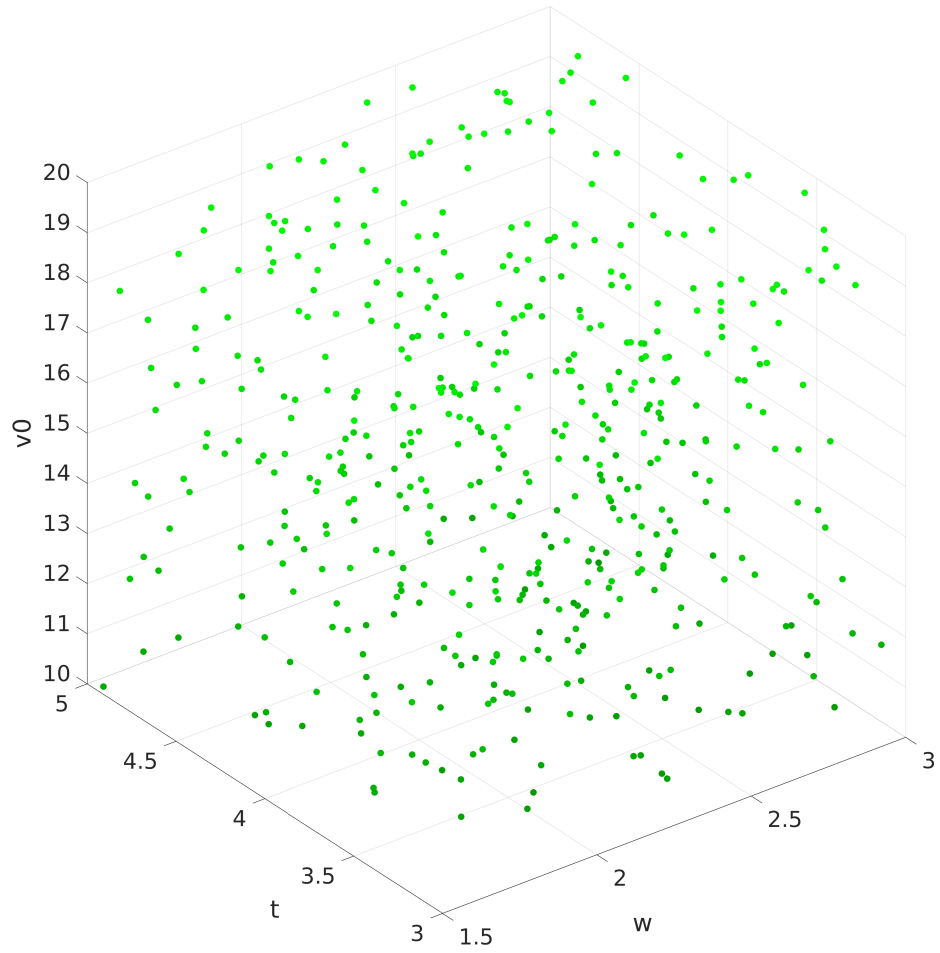


Figure 18: Scatter plot of the 500 evaluations of the random test cases for the single-lane change. Light-colored green points indicate lower robustnesses.

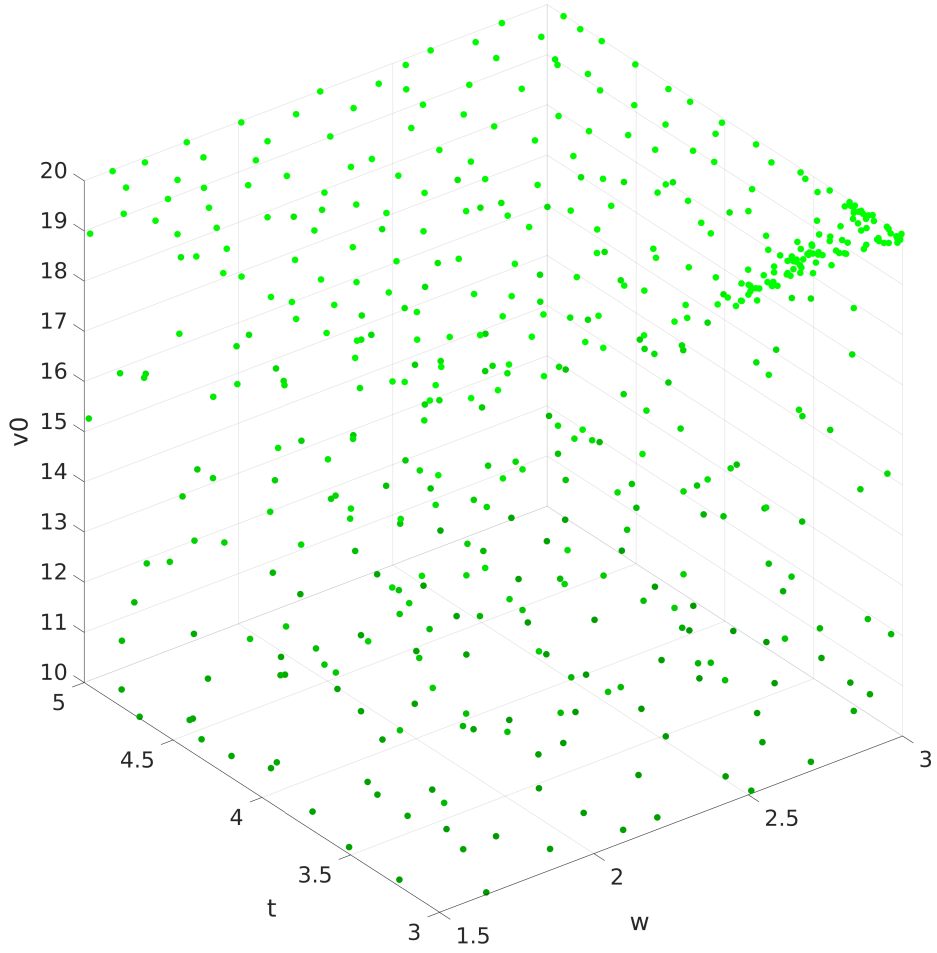


Figure 19: Scatter plot of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the single-lane change. Light-colored green points indicate lower robustnesses.

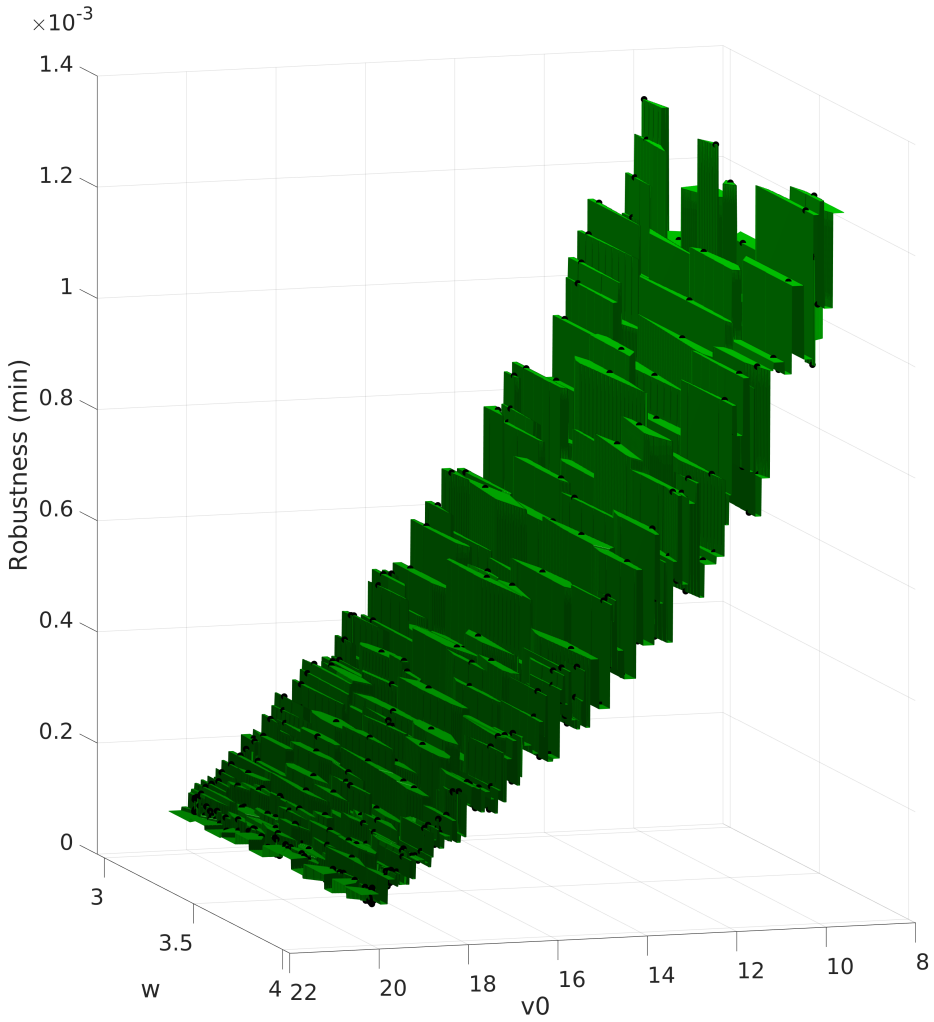


Figure 20: Voronoi plot (projection $(v0, w)$) of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the single-lane change.

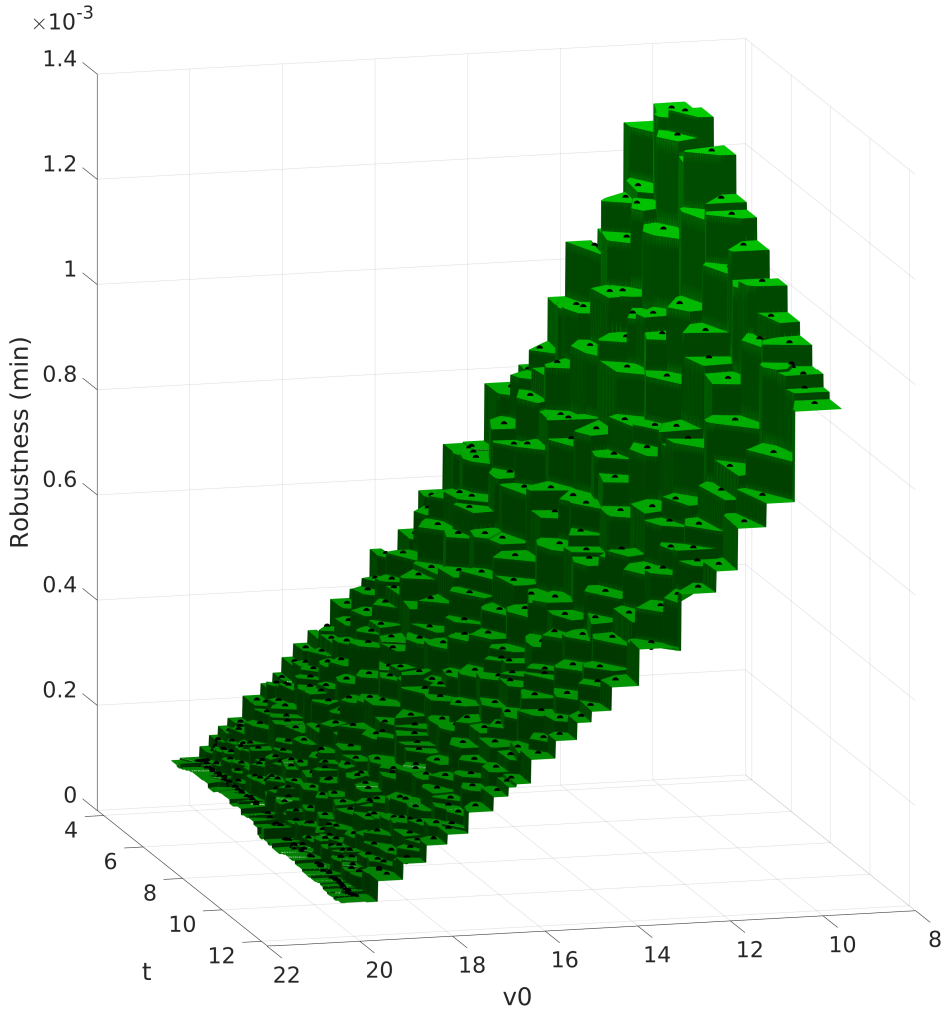


Figure 21: Voronoi plot (projection $(v0, t)$) of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the single-lane change.

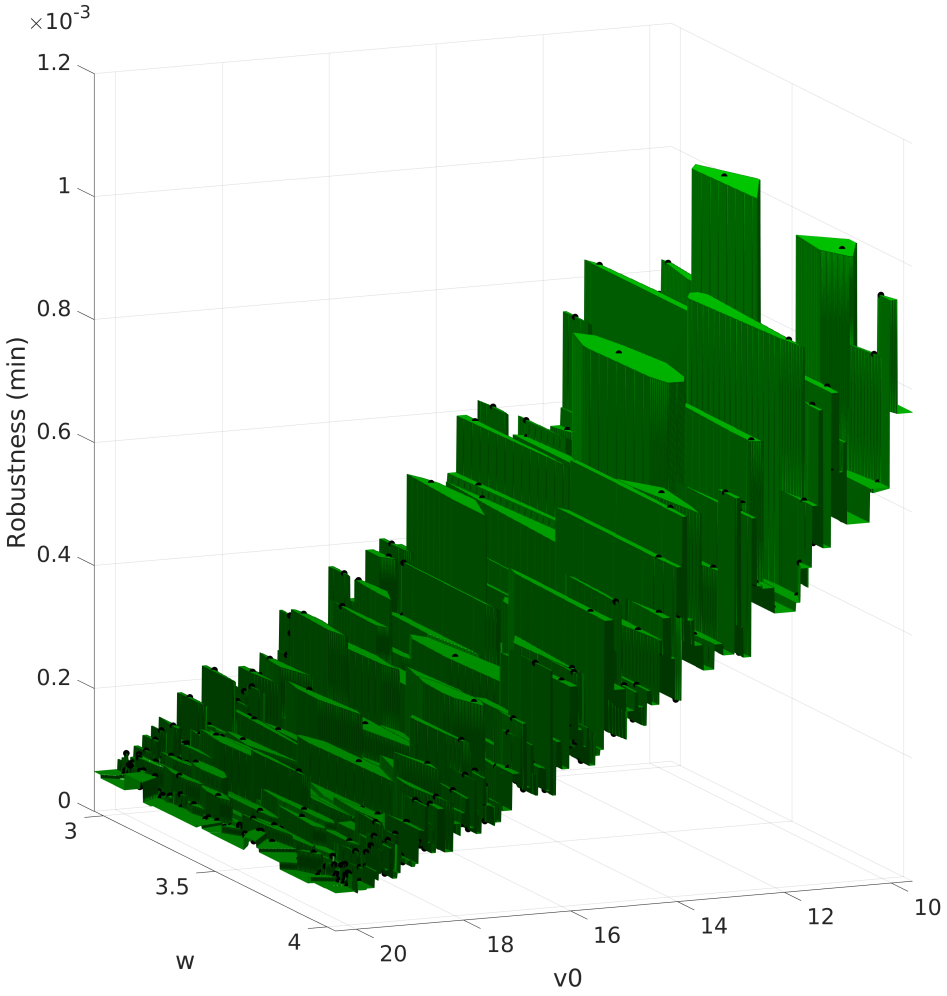


Figure 22: Voronoi plot (projection $(v0, w)$) of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the double-lane change.

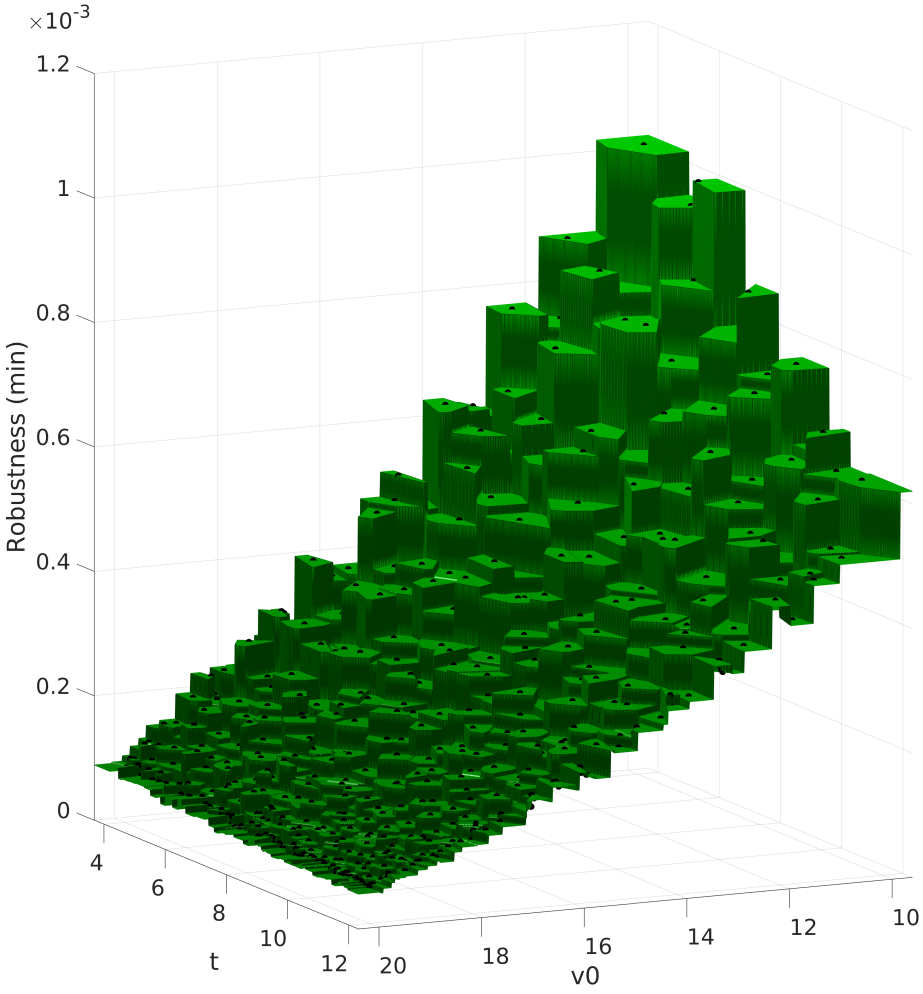


Figure 23: Voronoi plot (projection $(v0, t)$) of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the double-lane change.

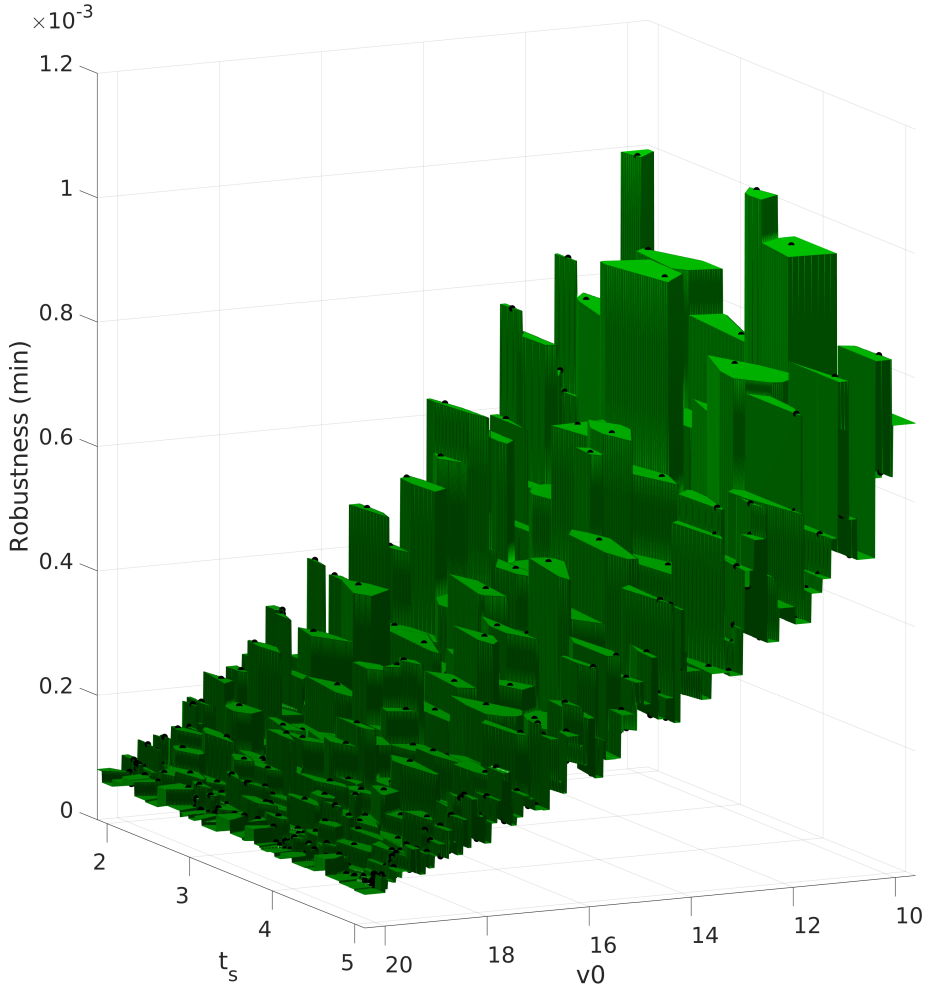


Figure 24: Voronoi plot (projection $(v0, t_s)$) of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the double-lane change.

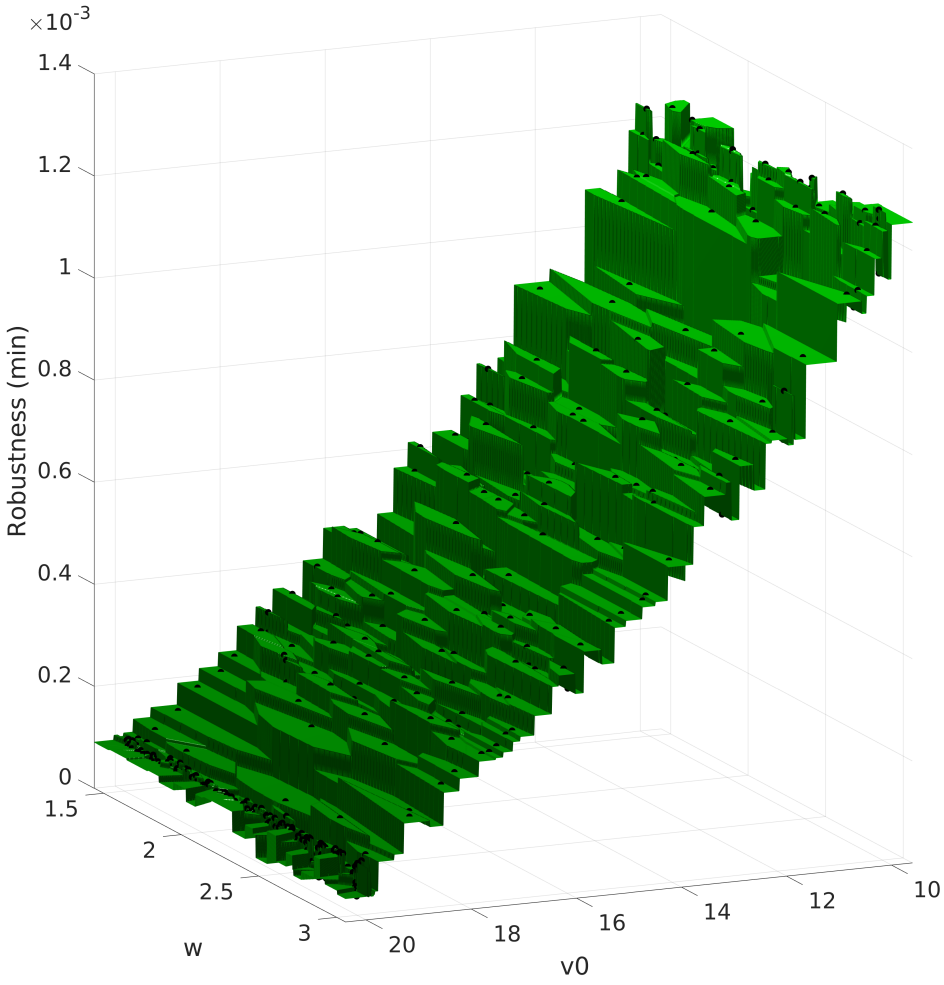


Figure 25: Plot (projection $(v0, w)$) of the robustness values of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the swerve maneuver.

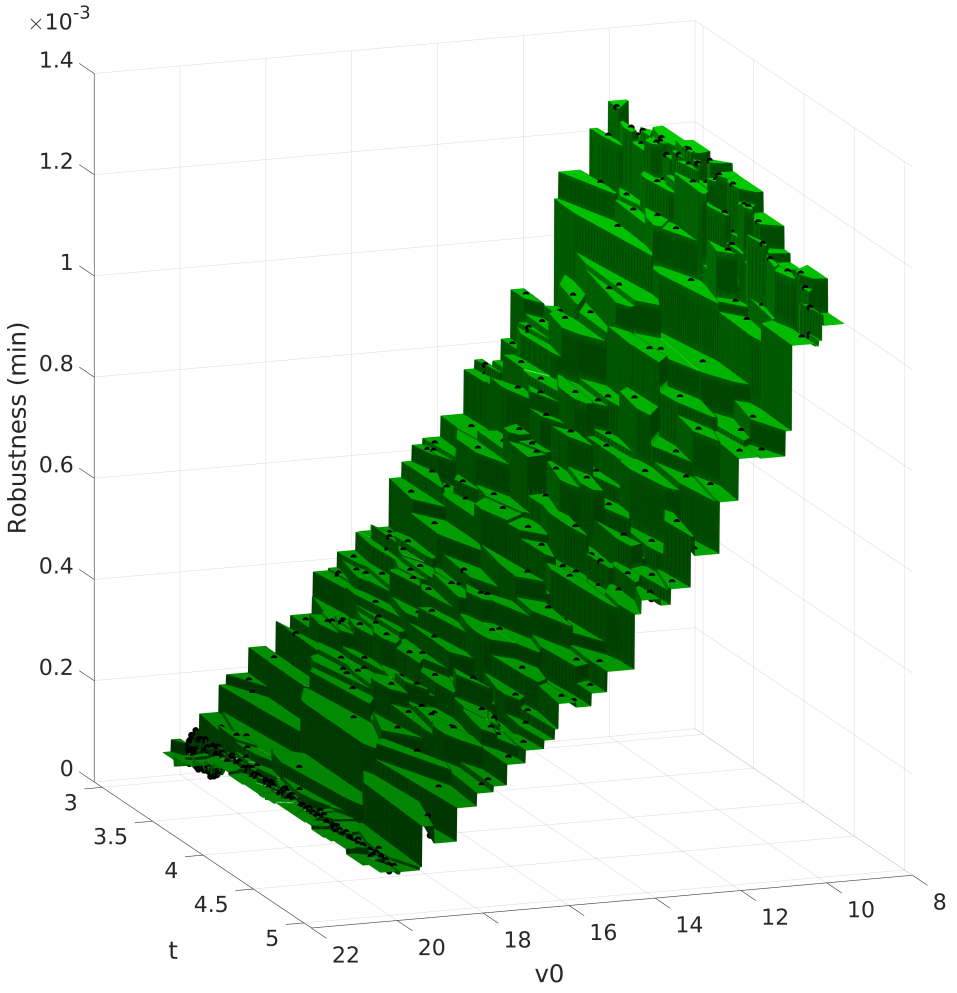


Figure 26: Plot (projection $(v0, t)$) of the robustness values of the 500 evaluations of the search-based reachset conformance testing approach using the Bayesian optimizer for the swerve maneuver.

We also investigated the swerve maneuver regarding coverage as a test end criterion and highlight the challenges as described in Section 4.3.

Table 11 summarizes the values of (14) for different Gaussian kernel parameters. The integral in (14) describes the average probability that a uniformly random selection of a test case fails according to the Gaussian process model. We mentioned before, that a Gaussian regression is not unique and depends on a number of hyperparameters.

In Table 11 we choose different Gaussian kernel parameters to illustrate the impact on a potential test end criterion based on the test cases from the search-based testing approach for the swerve maneuver. As toolbox, we used MATLAB 2018a's Statistics and Machine Learning Toolbox with kernel function "ardexponential" in the Gaussian process and for numerical integration in (14) the About Guaranteed Automatic Integration Library (GAIL) [37]. We first normalized each dimension of the parameter space by scaling it with the size of the interval for the parameter bounds. We then varied the signal standard deviation kernel parameter for the Gaussian regression. Intuitively speaking, this hyperparameter describes the increase of uncertainty in the Gaussian process model with increasing Euclidean distance in the normalized parameter space. In other words, even in the presence of only positive tests, the estimated probability of finding a negative test will increase, as the Gaussian process will assign a higher level of uncertainty to unexplored areas of the test space.

We altered this parameter to investigate the impact on (14). Furthermore, Figure 27 shows a plot in projection (w, v_0) of the standard deviation σ_p^2 for the different kernel parameter of the signal standard deviations of $3 \cdot 10^{-4}$ and 10^{-5}. Note that due to the numerics in the GP regression and the numerics within plotting of Figure 27, the ends of the spikes are not equal to zero as they theoretically should be for parameter choices equal to actual test cases. Figure 28 shows the values of $P(r < 0|p)$ (see (13)) that are used in (14). Here, we did not plot the values with signal standard deviation 10^{-5} as these are all zero.

Our numerical experiments show, as mentioned before in Section 4.3, that choosing kernel parameters in the GP regression has a direct influence on the values of the standard deviation σ_p^2 (see Figure 27) and the probability $P(r < 0|p)$ (see Figure 28) and thus the values of the integral in (14). If these parameters are not chosen with care, one would be mislead. Exaggerating, with a signal standard deviation of 10^{-5}, the 500 test cases would be enough to push the average probability that a uniformly random selection of a test case fails arbitrary close to zero.

Note that, due to the continuity of the physical models we used for conformance testing, the Gaussian process regressions reflected the true landscapes rather well. This is again in contrast to black-box testing problems where one could expect a lot of discontinuities in the

Table 11: Overview of the values computed for (14) based on different parameters in the Gaussian process model for the 500 test cases of the search-based testing approach for the swerve maneuver.

Normalization factor	Signal standard deviation	Value of Eq. (14)
parameter bounds of swerve maneuver	10^0	50%
parameter bounds of swerve maneuver	$3 \cdot 10^{-4}$	1%
parameter bounds of swerve maneuver	10^{-5}	0%

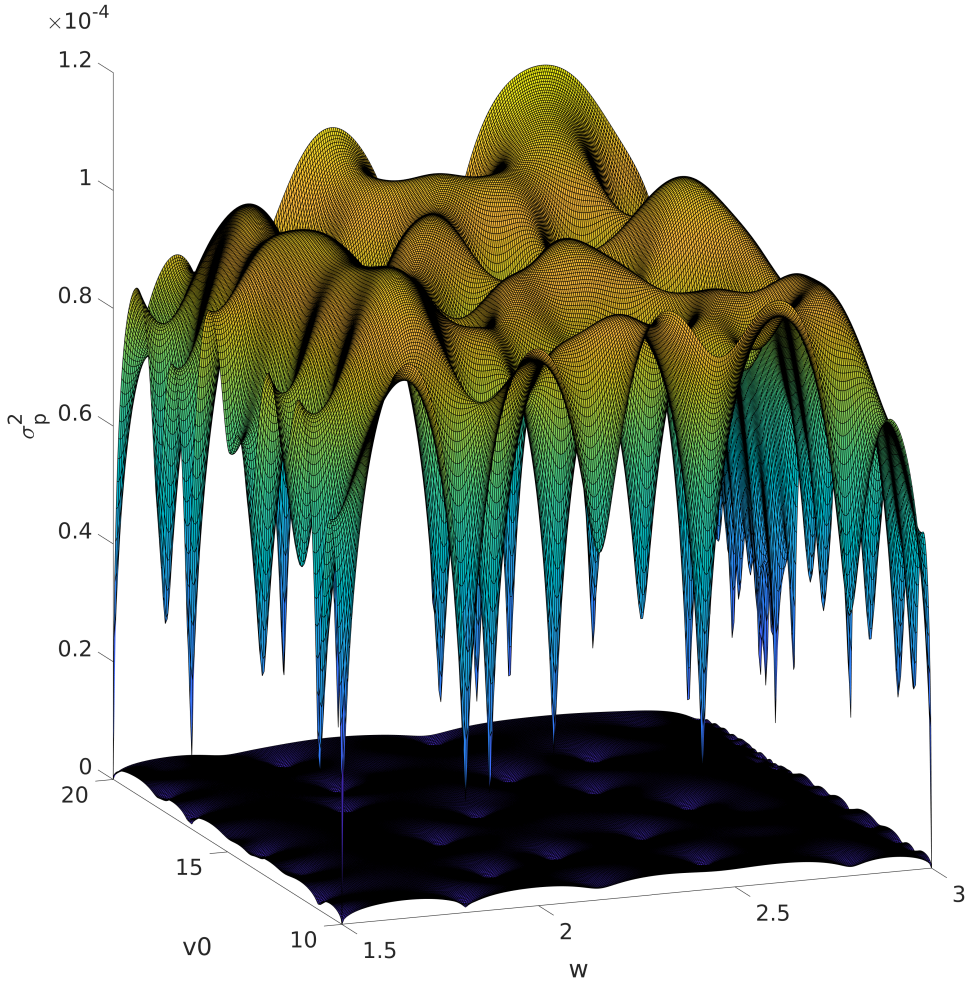


Figure 27: Plot (projection (w, v_0) for constant $t = 3$) of the standard deviation σ_p^2 within GP regressions for signal standard deviation $3 \cdot 10^{-4}$ (above) and 10^{-5} (below) based on the test cases from the search-based testing approach of the swerve maneuver.

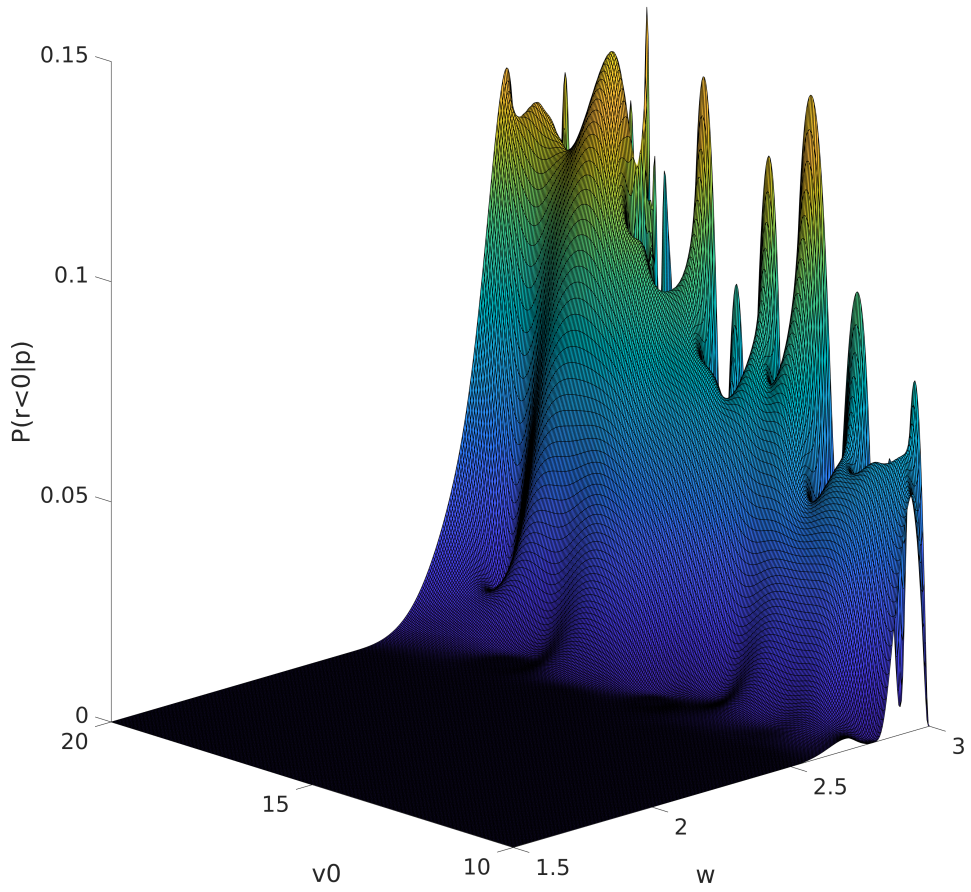


Figure 28: Plot (projection (w, v_0) for constant $t = 3$) of $P(r < 0|p)$ (according to (13)) for the signal standard deviation $3 \cdot 10^{-4}$ based on the test cases from the search-based testing approach of the swerve maneuver. Note that we did not plot the values with signal standard deviation 10^{-5} as these are all zero.

robustness landscape, as for example the planner part of the software would also be included. Nevertheless, this work is only a starting point, as physical models with closed loop controller will often also exhibit some switching which needs to be captured by a regression model.

Also, as we illustrated, the quantitative measure APSV is highly dependent on the regression hyperparameters. While it seems that suitable choices for these could be argued from assumptions on the physical system (e.g., bounds on parameter sensitivities), the precise definition of such hyperparameters is still an open problem for the automated driving use case.

6 Conclusions

In this deliverable we presented a survey conformance relations for cyber-physical systems in Section 2. The survey presented definitions of conformance relations, what properties transfer with conformance, and what which conformance verification and conformance testing methods exist.

Section 3 discussed how conformance testing as used within in UnCoVerCPS relates to the development process from an automotive perspective. In summary, conformance testing latches on existing process steps, however there is an increase in complexity as models and verification techniques have to include non-deterministic uncertainties in a set-based fashion.

Within Section 4 we introduced a novel algorithm to identify reachset conformant models. Furthermore, we introduced an approach for automated reachset conformance test case generation. In order to decide when to stop testing, we presented a novel approach for test coverage as a test end criterion with focus on continuous dynamics.

Section 5 presented experimental results for the approaches introduced in Section 4. We showed that our novel algorithm is able to identify reachset conformant vehicle models for all recorded measurements from real experiments of the DLR vehicle. Thereby, the identified models are not too conservative and thus likely to be of use within UnCoVerCPS online verification approach.

The numerical experiments for test case generation revealed that, compared to random testing, the proposed search-based testing approach for reachset conformance is more likely to find more crucial test cases within the same amount of test cases. Furthermore, our experiments indicate that identifying robust conformant models might be reducible to an identification on the corner cases of the input parameter space. Thus, effort in the development process could be reduced. This is a clear advantage of the UnCoVerCPS approach over the state of the art in industry, as the formal nature of the UnCoVerCPS controller leads to simpler, decomposed testing problems on individual physical models. Our experiments indicate that a significant reduction of testing effort is possible in this manner.

Regarding test coverage based on Gaussian surrogate models for use within a test end criterion, our experiments highlighted the necessity to research and discuss the “right” hyperparameters within the presented approach. Furthermore, additional political and societal discussions as well as statistical arguments are required to result in some kind of norm or standard, which is still very much work in progress.

References

[1] UnCoVerCPS Deliverable D5.2: Report on conformance testing of application models. Technical report.

[2] Vires - virtual test drive: https://vires.com/vtd-vires-virtual-test-drive/.

[3] A. Abate. Approximation metrics based on probabilistic bisimulations for general state-space markov processes: a survey. *Electronic Notes in Theoretical Computer Science*, 297:3–25, 2013.

[4] H. Abbas and G. Fainekos. Towards composition of conformant systems. *arXiv preprint arXiv:1511.05273*, 2015.

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

[6] H. Abbas, H. D. Mittelmann, and G. E. Fainekos. Formal property verification in a conformance testing framework. In *Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE 2014, Lausanne, Switzerland, October 19-21, 2014*, pages 155–164. IEEE, 2014.

[7] A. Aerts, M. R. Mousavi, and M. A. Reniers. A tool prototype for model-based testing of cyber-physical systems. In M. Leucker, C. Rueda, and F. D. Valencia, editors, *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings*, volume 9399 of *Lecture Notes in Computer Science*, pages 563–572. Springer, 2015.

[8] A. Aerts, M. Reniers, and M. Mousavi. Chapter 19 - model-based testing of cyber-physical systems. In H. Song, D. B. Rawat, S. Jeschke, and C. Brecher, editors, *Cyber-Physical Systems*, Intelligent Data-Centric Systems, pages 287 – 304. Academic Press, Boston, 2017.

[9] B. K. Aichernig, H. Brandl, E. Jöbstl, and W. Krenn. Model-based mutation testing of hybrid systems. In F. S. de Boer, M. M. Bonsangue, S. Hallerstede, and M. Leuschel, editors, *Formal Methods for Components and Objects - 8th International Symposium, FMCO 2009, Eindhoven, The Netherlands, November 4-6, 2009. Revised Selected Papers*, volume 6286 of *Lecture Notes in Computer Science*, pages 228–249. Springer, 2009.

[10] B. K. Aichernig, H. Brandl, and F. Wotawa. Conformance testing of hybrid systems with qualitative reasoning models. *Electronic Notes in Theoretical Computer Science*, 253(2):53–69, 2009.

[11] B. K. Aichernig, F. Lorber, and D. Nickovic. Time for mutants - model-based mutation testing with timed automata. In M. Veanes and L. Viganò, editors, *Tests and Proofs - 7th International Conference, TAP 2013, Budapest, Hungary, June 16-20, 2013. Proceedings*, volume 7942 of *Lecture Notes in Computer Science*, pages 20–38. Springer, 2013.

[12] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[13] M. Althoff and J. M. Dolan. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1162–1167. IEEE, 2011.

[14] M. Althoff and J. M. Dolan. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *American Control Conference, ACC 2012, Montreal, QC, Canada, June 27-29, 2012*, pages 3559–3566. IEEE, 2012.

[15] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[16] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional refinement for hierarchical hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 33–48. Springer, 2001.

[17] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional modeling and refinement for hierarchical hybrid systems. *The Journal of Logic and Algebraic Programming*, 68(1-2):105–128, 2006.

[18] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.

[19] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[20] Y. S. R. Annapureddy and G. E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proc. of the 36th Annual Conference of IEEE Industrial Electronics*, pages 91–96, 2010.

[21] D. Araiza-Illan, D. Western, A. Pipe, and K. Eder. Coverage-driven verification: An approach to verify code for robots that directly interact with humans. In *Proc. of the 11th International Hardware and Software: Verification and Testing*, pages 69–84, 2015.

[22] D. Araiza-Illan, D. Western, A. Pipe, and K. Eder. Systematic and realistic testing in simulation of control code for robots in collaborative human-robot interactions. In *Towards Autonomous Robotic Systems: 17th Annual Conference*, pages 20–32, 2016.

[23] D. Araiza-Illan, D. Western, A. Pipe, and K. Eder. Systematic and realistic testing in simulation of control code for robots in collaborative human-robot interactions. In *Towards Autonomous Robotic Systems: 17th Annual Conference*, pages 20–32, 2016.

[24] H. Araujo, G. Carvalho, A. Sampaio, M. R. Mousavi, and M. Taromirad. A process for sound conformance testing of cyber-physical systems. In *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 46–50, March 2017.

[25] R. Back and J. von Wright. *Refinement Calculus - A Systematic Introduction.* Graduate Texts in Computer Science. Springer, 1998.

[26] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series).* The MIT Press, 2008.

[27] R. Banach, H. Zhu, W. Su, and X. Wu. Continuous asm, and a pacemaker sensing fragment. In *International Conference on Abstract State Machines, Alloy, B, VDM, and Z*, pages 65–78. Springer, 2012.

[28] O. Beg, H. Abbas, T. T. Johnson, and A. Davoudi. Model validation of PWM DC-DC converters. *IEEE Trans. Industrial Electronics*, 64(9):7049–7059, 2017.

[29] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. Property preserving simulations. In G. von Bochmann and D. K. Probst, editors, *Computer Aided Verification, Fourth International Workshop, CAV '92, Montreal, Canada, June 29 - July 1, 1992, Proceedings*, volume 663 of *Lecture Notes in Computer Science*, pages 260–273. Springer, 1992.

[30] G. Bian and A. Abate. On the relationship between bisimulation and trace equivalence in an approximate probabilistic context. In *International Conference on Foundations of Software Science and Computation Structures*, pages 321–337. Springer, 2017.

[31] H. Brandl, G. Fraser, and F. Wotawa. Coverage-based testing using qualitative reasoning models. In *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1-3, 2008*, pages 393–398. Knowledge Systems Institute Graduate School, 2008.

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

[33] M. L. Bujorianu, J. Lygeros, and M. C. Bujorianu. Bisimulation for general stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 198–214. Springer, 2005.

[34] A. Casagrande. Hybrid automata and bisimulations. EUT Edizioni Università di Trieste, 2010.

[35] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of Computer-Aided Verification*, LNCS 8044, pages 258–263. Springer, 2013.

[36] J. J. Chilenski and S. P. Miller. Applicability of modified condition/decision coverage to software testing. *Software Engineering Journal*, 9(5):193–200, Sep. 1994.

[37] S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, L. D. Jiménez Rugama, Lluís Antoni, J. Rathinavel, K. Zhang, X. Tong, Y. Zhang, and X. Zhou. GAIL: Guaranteed Automatic Integration Library (Version 2.2), MATLAB Software, 2017.

[38] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE transactions on software engineering*, (3):178–187, 1978.

[39] D. Chu and D. D. Siljak. A canonical form for the inclusion principle of dynamic systems. *SIAM Journal on Control and Optimization*, 44(3):969–990, 2005.

[40] P. J. Cuijpers. On bicontinuous bisimulation and the preservation of stability. In *International Workshop on Hybrid Systems: Computation and Control*, pages 676–679. Springer, 2007.

[41] T. Dang. Model-based testing of hybrid systems. In J. Zander, I. Schieferdecker, and P. J. Mosterman, editors, *Model-Based Testing for Embedded Systems*, chapter 14, pages 383–424. CRC Press, Inc., 2011.

[42] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[43] T. Dang and T. Nahhal. Model-based testing of hybrid systems. Technical report, Verimag, IMAG, Nov 2007.

[44] T. Dang and N. Shalev. Test coverage estimation using threshold accepting. In F. Cassez and J.-F. Raskin, editors, *Automated Technology for Verification and Analysis*, volume 8837 of *Lecture Notes in Computer Science*, pages 115–128. Springer International Publishing, 2014.

[45] J. V. Deshmukh, R. Majumdar, and V. S. Prabhu. Quantifying conformance using the skorokhod metric. In D. Kroening and C. S. Pasareanu, editors, *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, volume 9207 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2015.

[46] A. D'Innocenzo, A. Abate, and M. D. D. Benedetto. Approximate abstractions of discrete-time controlled stochastic hybrid systems. In *Proceedings of the 47th IEEE Conference on Decision and Control, CDC 2008, December 9-11, 2008, Cancún, México*, pages 221–226. IEEE, 2008.

[47] A. Donzé. *Trajectory-Based Verification and Controller Synthesis for Continuous and Hybrid Systems*. PhD thesis, University Joseph Fourier, 2007.

[48] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Proc. of Computer-Aided Verification*, pages 167–170, 2010.

[49] G. Frehse. *Compositional Verification of Hybrid Systems Using Simulation Relations*. PhD thesis, Radboud Universiteit Nijmegen, 2005.

[50] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2005.

[51] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: scalable verification of hybrid systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[52] G. Frehse, Z. Han, and B. Krogh. Assume-guarantee reasoning for hybrid i/o-automata by over-approximation of continuous interaction. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 479–484. IEEE, 2004.

[53] A. Girard. A composition theorem for bisimulation functions. *arXiv preprint arXiv:1304.5153*, 2013.

[54] A. Girard. *Computational Approaches to Analysis and Control of Hybrid Systems*. Nov. 2013. Habilitation.

[55] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *IFAC Proceedings Volumes*, 39(5):106–111, 2006.

[56] A. Girard, A. A. Julius, and G. J. Pappas. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems*, 18(2):163–179, 2008.

[57] A. Girard and G. Pappas. Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, 52(5):782–798, May 2007.

[58] A. Girard and G. J. Pappas. Approximate bisimulations for constrained linear systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 4700–4705. IEEE, 2005.

[59] A. Girard and G. J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 684–689. IEEE, 2005.

[60] A. Girard and G. J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307 – 1317, 2007.

[61] A. Girard and G. J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45(2):566–571, 2009.

[62] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2010.

[63] K. A. Grasse. Simulation and bisimulation of nonlinear control systems with admissible classes of inputs and disturbances. *SIAM J. Control Optim.*, 46(2):562–584, Apr. 2007.

[64] K. A. Grasse and N. Ho. Simulation relations and controllability properties of linear and nonlinear control systems. *SIAM Journal on Control and Optimization*, 53(3):1346–1374, 2015.

[65] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theor. Comput. Sci.*, 342(2-3):229–261, 2005.

[66] T. A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 278–292, 1996.

[67] T. A. Henzinger, R. Majumdar, and V. S. Prabhu. Quantifying similarities between timed systems. In P. Pettersson and W. Yi, editors, *Formal Modeling and Analysis of Timed Systems, Third International Conference, FORMATS 2005, Uppsala, Sweden, September 26-28, 2005, Proceedings*, volume 3829 of *Lecture Notes in Computer Science*, pages 226–241. Springer, 2005.

[68] T. A. Henzinger, M. Minea, and V. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 275–290. Springer, 2001.

[69] D. Heß, M. Althoff, and T. Sattel. Formal verification of maneuver automata for parameterized motion primitives. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1474–1481. IEEE, 2014.

[70] D. Heß, C. Löper, and T. Hesse. Safe cooperation of automated vehicles. In *AAET - Automatisiertes und vernetztes Fahren, Beitr"age zum gleichnamigen 18. Braunschweiger Symposium vom 8. und 9. Februar 2017*, pages 309–334. ITS automotive nord e.V., 2017.

[71] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. J. Krause, G. Lüttgen, A. J. H. Simons, S. A. Vilkomir, M. R. Woodward, and H. Zedan. Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2):9:1–9:76, 2009.

[72] N. Ho. *Controllability of Linear and Nonlinear Control Systems Related Through Simulation Relations*. PhD thesis, UNIVERSITY OF OKLAHOMA, 2015.

[73] M. Ikeda, D. Siljak, and D. White. An inclusion principle for dynamic systems. In *American Control Conference, 1982*, pages 884–892. IEEE, 1982.

[74] A. A. Julius. Approximate abstraction of stochastic hybrid automata. In J. P. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control, 9th International Workshop, HSCC 2006, Santa Barbara, CA, USA, March 29-31, 2006, Proceedings*, volume 3927 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2006.

[75] A. A. Julius, A. D'Innocenzo, M. D. D. Benedetto, and G. J. Pappas. Approximate equivalence and synchronization of metric transition systems. *Systems & Control Letters*, 58(2):94–101, 2009.

[76] A. A. Julius, A. Girard, and G. J. Pappas. Approximate bisimulation for a class of stochastic hybrid systems. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.

[77] A. A. Julius and G. J. Pappas. Approximations of stochastic hybrid systems. *IEEE Transactions on Automatic Control*, 54(6):1193–1203, 2009.

[78] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *Hybrid Systems: Computation and Control*, LNCS 2623, pages 283–297. Springer, 2003.

[79] N. Khakpour and M. R. Mousavi. Notions of Conformance Testing for Cyber-Physical Systems: Overview and Roadmap (Invited Paper). In L. Aceto and D. de Frutos Escrig, editors, *26th International Conference on Concurrency Theory (CONCUR 2015)*, volume 42 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18–40, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[80] X. D. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7):1026–1049, 2000.

[81] M. Krichen and S. Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.

[82] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines-a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[83] S. B. Liu, H. Roehm, C. Heinzemann, I. Lütkebohle, J. Oehlerking, and M. Althoff. Provably safe motion of mobile robots in human environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 1351–1357. IEEE, 2017.

[84] S. M. Loos and A. Platzer. Differential refinement logic. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 505–514. ACM, 2016.

[85] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata revisited. In M. D. D. Benedetto and A. L. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control, 4th International Workshop, HSCC 2001, Rome, Italy, March 28-30, 2001, Proceedings*, volume 2034 of *Lecture Notes in Computer Science*, pages 403–417. Springer, 2001.

[86] G. Ma, L. Qin, X. Liu, C. Shi, and G. Wu. Approximate bisimulations for constrained discrete-time linear systems. In *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, pages 1058–1063. IEEE, 2015.

[87] R. Majumdar and V. S. Prabhu. Computing the skorokhod distance between polygonal traces. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 199–208. ACM, 2015.

[88] R. Majumdar and V. S. Prabhu. Computing distances between reach flowpipes. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 267–276. ACM, 2016.

[89] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In Y. Lakhnech and S. Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*, volume 3253 of *Lecture Notes in Computer Science*, pages 152–166. Springer, 2004.

[90] I. M. Mitchell. Comparing forward and backward reachability as tools for safety analysis. In A. Bemporad, A. Bicchi, and G. C. Buttazzo, editors, *Hybrid Systems: Computation and Control, 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007, Proceedings*, volume 4416 of *Lecture Notes in Computer Science*, pages 428–443. Springer, 2007.

[91] S. Mitsch and A. Platzer. Modelplex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49(1-2):33–74, 2016.

[92] S. Mitsch, J.-D. Quesel, and A. Platzer. Refactoring, refinement, and reasoning. In *International Symposium on Formal Methods*, pages 481–496. Springer, 2014.

[93] M. Mohaqeqi and M. Mousavi. Towards an approximate conformance relation for hybrid i/o automata. In *Proceedings of the 1st International Workshop on Verification and Validation of Cyber-Physical Systems (V2CPS)*, 2016.

[94] M. Mohaqeqi and M. R. Mousavi. Sound test-suites for cyber-physical systems. In *10th International Symposium on Theoretical Aspects of Software Engineering, TASE 2016, Shanghai, China, July 17-19, 2016*, pages 42–48. IEEE Computer Society, 2016.

[95] M. Mohaqeqi, M. R. Mousavi, and W. Taha. Conformance testing of cyber-physical systems: A comparative study. *ECEASST*, 70, 2014.

[96] L. Munteanu and K. A. Grasse. Constructing simulation relations for ido systems affine in inputs and disturbances. *Mathematics of Control, Signals, and Systems*, 27(3):317–346, 2015.

[97] A. Murthy, M. A. Islam, E. Bartocci, E. M. Cherry, F. H. Fenton, J. Glimm, S. A. Smolka, and R. Grosu. Approximate bisimulations for sodium channel dynamics. In *Computational Methods in Systems Biology*, pages 267–287. Springer, 2012.

[98] A. Murthy, M. A. Islam, S. A. Smolka, and R. Grosu. Computing bisimulation functions using sos optimization and δ-decidability over the reals. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 78–87. ACM, 2015.

[99] A. Murthy, M. A. Islam, S. A. Smolka, and R. Grosu. Computing compositional proofs of input-to-output stability using sos optimization and δ-decidability. *Nonlinear Analysis: Hybrid Systems*, 23:272–286, 2017.

[100] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivančić, A.Gupta, and G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Hybrid Systems: Computation and Control*, pages 211–220, 2010.

[101] H. Pan, M. Zhang, and Y. Chen. Approximate simulation for metric hybrid input/output automata. In *Secure Software Integration & Reliability Improvement Companion (SSIRI-C), 2011 5th International Conference on*, pages 53–59. IEEE, 2011.

[102] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

[103] A. Platzer and J.-D. Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *International Joint Conference on Automated Reasoning*, pages 171–178. Springer, 2008.

[104] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[105] G. Pola, A. J. Van Der Schaft, and M. D. Di Benedetto. Bisimulation theory for switching linear systems. 2004.

[106] P. Prabhakar, G. Dullerud, and M. Viswanathan. Pre-orders for reasoning about stability. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 197–206. ACM, 2012.

[107] P. Prabhakar, G. Dullerud, and M. Viswanathan. Stability preserving simulations and bisimulations for hybrid systems. *IEEE Transactions on Automatic Control*, 60(12):3210–3225, 2015.

[108] P. Prabhakar and J. Liu. Bisimulations for input-output stability of hybrid systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5515–5520, Dec 2016.

[109] V. Preoteasa and S. Tripakis. Towards compositional feedback in non-deterministic and non-input-receptive systems. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 768–777. ACM, 2016.

[110] J.-D. Quesel. *Similarity, Logic, and Games: Bridging Modeling Layers of Hybrid Systems*. PhD thesis, 2013.

[111] J.-F. Raskin. *Logics, automata and classical theories for deciding real time*. PhD thesis, Facultés universitaires Notre-Dame de la Paix, Namur, 1999.

[112] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[113] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, and E. H. andTobias Nipkow. Formalising and monitoring traffic rules for autonomous vehicles in isabelle/hol. In N. Polikarpova and S. Schneider, editors, *integrated Formal Methods (iFM 2017)*, volume 10510, pages 50–66, 2017.

[114] H. Roehm, T. Heinz, and E. C. Mayer. Stlinspector: STL validation with guarantees. In R. Majumdar and V. Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 225–232. Springer, 2017.

[115] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. STL model checking of continuous and hybrid systems. In C. Artho, A. Legay, and D. Peled, editors, *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings*, volume 9938 of *Lecture Notes in Computer Science*, pages 412–427, 2016.

[116] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Reachset conformance testing of hybrid automata. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016, Vienna, Austria, April 12-14, 2016*, pages 277–286, 2016.

[117] M. Roggenbach and M. Majster-Cederbaum. Towards a unified view of bisimulation: a comparative study. *Theoretical Computer Science*, 238(1):81 – 130, 2000.

[118] B. S. Rüffer, C. M. Kellett, and S. R. Weller. Integral input-to-state stability of interconnected iiss systems by means of a lower-dimensional comparison system. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 638–643. IEEE, 2009.

[119] J. Schmaltz and J. Tretmans. On conformance testing for timed systems. In F. Cassez and C. Jard, editors, *Formal Modeling and Analysis of Timed Systems, 6th International Conference, FORMATS 2008, Saint Malo, France, September 15-17, 2008. Proceedings*, volume 5215 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2008.

[120] G. V. Smirnov. *Introduction to the Theory of Differential Inclusions*. American Mathematical Society, 2002.

[121] A. M. Stanković, S. D. Dukić, and A. T. Sarić. Approximate bisimulation-based reduction of power system dynamic models. *IEEE Transactions on Power Systems*, 30(3):1252–1260, 2015.

[122] T. Strathmann and J. Oehlerking. Experience report: Verifying properties of an electro-mechanical braking system. *ARCH*, 2015.

[123] P. Tabuada. Approximate simulation relations and finite abstractions of quantized control systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 529–542. Springer, 2007.

[124] P. Tabuada. *Verification and Control of Hybrid Systems - A Symbolic Approach*. Springer, 2009.

[125] P. Tabuada and G. J. Pappas. Bisimilar control affine systems. *Systems & Control Letters*, 52(1):49–58, 2004.

[126] P. Tabuada, G. J. Pappas, and P. Lima. Compositional abstractions of hybrid control systems. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 352–357. IEEE, 2001.

[127] P. Tabuada, G. J. Pappas, and P. Lima. Compositional abstractions of hybrid control systems. *Discrete Event Dynamic Systems*, 14(2):203–238, 2004.

[128] H. Tanner and G. J. Pappas. Simulation relations for discrete-time linear systems. *IFAC Proceedings Volumes*, 35(1):445–450, 2002.

[129] H. G. Tanner and G. J. Pappas. Abstractions of constrained linear systems. In *American Control Conference, 2003. Proceedings of the 2003*, volume 4, pages 3381–3386. IEEE, 2003.

[130] S. Tasiran. *Compositional and hierarchical techniques for the formal verification of real-time systems*. PhD thesis, Citeseer, 1998.

[131] G. J. Tretmans. *A formal approach to conformance testing*. PhD thesis, Universiteit Twente, 1992.

[132] J. Tretmans. Testing concurrent systems: A formal approach. In J. C. M. Baeten and S. Mauw, editors, *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings*, volume 1664 of *Lecture Notes in Computer Science*, pages 46–65. Springer, 1999.

[133] A. Van Der Schaft. Bisimulation of dynamical systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 555–569. Springer, 2004.

[134] A. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Trans. Automat. Contr.*, 49(12):2160–2172, 2004.

[135] M. Van Osch. Hybrid input-output conformance and test generation. In *Formal Approaches to Software Testing and Runtime Verification*, pages 70–84. Springer, 2006. appended file is a more detailed technical report.

[136] M. P. W. J. van Osch. *Automated model-based testing of hybrid systems*. PhD thesis, Eindhoven University of Technology, 2009.

[137] C. Wang, J. Wu, H. Tan, and J. Fu. Approximate reachability and bisimulation equivalences for transition systems. *Transactions of Tianjin University*, 22:19–23, 2016.

[138] G. Yan, L. Jiao, Y. Li, S. Wang, and N. Zhan. Approximate bisimulation and discretization of hybrid csp. In *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*, pages 702–720. Springer, 2016.

[139] K. Yang and H. Ji. Hierarchical analysis of large-scale control systems via vector simulation function. *Systems & Control Letters*, 102:74 – 80, 2017.