Model Conformance for Cyber-Physical Systems: A Survey

HENDRIK ROEHM, Robert Bosch GmbH
JENS OEHLERKING, Robert Bosch GmbH
MATTHIAS WOEHRLE, Robert Bosch GmbH
MATTHIAS ALTHOFF, TU Munich

Model-based development is an important paradigm for developing cyber-physical systems (CPS). The underlying assumption is that the functional behavior of a model is related to the behavior of a more concretized model or the real system. A formal definition of such a relation is called conformance relation. There are a variety of conformance relations, and the question arises of how to select a conformance relation for the development of CPS. The contribution of this paper is a survey of the definitions and algorithms of conformance relations for CPS. Additionally, the paper compares several conformance relations and provides guidance on which relation to select for specific problems. Finally, we discuss how to select inputs to test conformance.

CCS Concepts: • **Computing methodologies** → **Model development and analysis**; • **Software and its engineering** → *Formal software verification*;

Additional Key Words and Phrases: Cyber-physical systems, hybrid systems, verification, conformance testing, reachset conformance, input-output conformance simulation, bisimulation, conformance relation, hybrid input-output conformance, hioco, language inclusion, refinement, implementation relation, behavioral inclusion, projective relational refinement, trace conformance, trace equivalence, language equivalence, input-output equivalence, continuous simulation, approximate trace conformance, approximate language inclusion, tau-epsilon closeness, epsilon-delta similarity, skorokhod conformance, approximate simulation, approximate bisimulation, qrioco, qualtitative reasoning input output conformance.

ACM Reference format:
Hendrik Roehm, Jens Oehlerking, Matthias Woehrle, and Matthias Althoff. 2019. Model Conformance for Cyber-Physical Systems: A Survey. *ACM Transactions on Cyber-Physical Systems* 1, 1, Article 1 (January 2019), 28 pages.
https://doi.org/10.1145/3306157

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

The authors gratefully acknowledge financial support by the European Commission project UnCoVerCPS under grant number 643921.

XXXX-XXXX/2019/1-ART1 $15.00
https://doi.org/10.1145/3306157

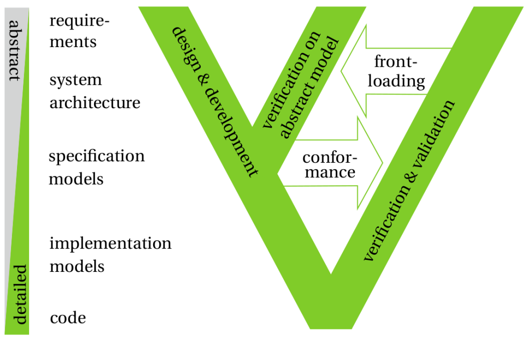

Fig. 1. The V-process characterization of model-based design.

discrete and continuous dynamics – so-called hybrid systems. Discrete and continuous models have severe differences – not only in their syntax and semantics – but also in the principles used to obtain those models. An example of a hybrid system with tight interconnections between discrete and continuous parts is an electro-mechanical brake [122].

The V-process characterizes the different development steps of 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, a detailed specification, and implementation models. Finally, this leads to the implemented code, shown at the bottom of the V. In the end, the implementation has to fulfill the requirements. In a second phase, the system is verified and validated, starting from the code level (e.g., by conducting unit tests of code) all the way up to the requirements. From the unit tests 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 abstraction levels. In every step, implementation aspects, such as scheduling of computations, quantization of variables, and sensor/actuator inaccuracies, are added as shown in Figure 2. In order to reduce the testing effort on the actual cyber-physical system, it is desirable to conduct as much testing as possible on models throughout the development process. However, in order to achieve this, test results on more abstract models used 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: early 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 verification methods in the cyber-physical systems 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 abstraction

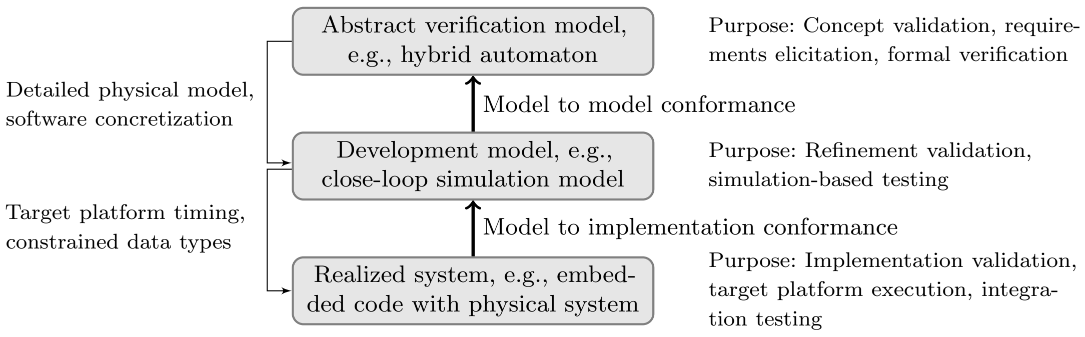
Abstract verification model, e.g., hybrid automaton
Purpose: Concept validation, requirements elicitation, formal verification
Detailed physical model, software concretization
Model to model conformance
Development model, e.g., close-loop simulation model
Purpose: Refinement validation, simulation-based testing
Target platform timing, constrained data types
Model to implementation conformance
Realized system, e.g., embedded code with physical system
Purpose: Implementation validation, target platform execution, integration testing

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

levels. We call these formal links *conformance relations*. In this paper, we survey different conformance relations proposed in the literature for cyber-physical systems, their transference properties, and approaches for systematic testing. As conformance relations for discrete and timed systems have already been studied in detail in the literature [25, 73, 117, 119], we focus on the conformance relations explicitly considering the continuous and hybrid part (see Section 3).

A variety of different conformance relations for cyber-physical systems have been proposed in the literature. Aerts et al. [6] provide a brief overview of model-based testing and conformance relations for hybrid systems. However, that overview does not attempt to be a comprehensive survey. The goal of our survey is to provide a detailed discussion about existing notions of conformance, which does not yet exist in the literature, to our best knowledge. Besides discussing the definitions and properties of conformance relations, we provide references to algorithmic approaches to check conformance and outline the main application areas.

This 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, 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 concepts.

The remainder is structured as follows: We first introduce the formal basis for conformance and review conformance for discrete systems in Section 2. Then, we give an overview of conformance relations and review work on conformance in the domain of cyber-physical systems in Section 3. 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 4. Finally, we discuss important aspects of input selection approaches for conformance testing in Section 5 and draw conclusions in Section 6. In each section, we also summarize existing work for discrete and timed systems.

2 PRELIMINARIES

Discrete systems can be modeled as transition systems which produce a sequence of discrete events. Timed systems extend discrete systems with time; passage of time can be considered as an additional real-valued event. Hybrid systems extend timed systems by allowing arbitrary continuous behavior. As hybrid systems are a superclass of both discrete and timed systems, the traces of hybrid systems can be abstracted to timed or discrete sequences, and all techniques and methods applicable for discrete and timed systems can be applied to hybrid systems as well. However, sufficiently accurate abstractions to discrete systems often result in an unmanagably large number of discrete states.

As opposed to discrete and timed systems, states and outputs of the physical part of hybrid systems are real-valued. A richer set of traces (real values over continuous time) is possible, compared to discrete sequences of discrete outputs. In particular, these CPS models are expected to react to inputs in dense time and to produce a dense-time output. A system model that does not define the reaction to a legitimate input at all times will typically be considered to be defective. Therefore, we focus in this survey on input-receptive systems; for a study on non-input-receptiveness,¡w see [111].

For modeling evolutions of a hybrid system, we follow Dang [43] and consider $inputs(S)$ as the set of piecewise continuous input functions of system S. A state of a hybrid system is 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 and $x(t) = (q_i, x_i(t))$ for $t \in [t_i, t_{t+1}]$. We can only observe output states, which are projections of states via the output map *out*. Contrary to a (white box) abstract model with formalized differential equations, a (black box) implementation typically can only be measured via its input/output behavior. 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 $otraces(S, u(.))$. In this survey, we consider non-deterministic systems, which means that multiple output traces are possible for the same input function. Without loss of generality, we assume that non-determinism is not modeled by non-deterministic inputs, but by differential inclusions [120]. In the hybrid systems community, Zeno[1] behavior is often assumed to be absent from the model or Zeno runs are not considered to be valid traces. Since Zeno behavior is a concept that cannot be observed in physical systems, it is often of little use in models as well; thus, we assume in this paper that all traces are non-Zeno.

Inspired by Tretmans [131], we formalize conformance 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 behaviors. A system S_A is correct with respect to *spec*, which we write as $S_A \models spec$, if the output behavior of S_A is a subset of the correct output behavior of *spec* (for the same inputs). A useful conformance relation `conf` between systems S_I and S_A implies transference for related systems, which is that all specified properties transfer from S_A to S_I

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

There exist different names for conformance relations, such as implementation relation [25] and refinement relation [19]. If a conformance relation holds, S_A is called an abstraction of S_I, and S_I is called a refinement of S_A [25]. For a conformance relation, it is important

[1]Infinitely many discrete transitions in finite time.

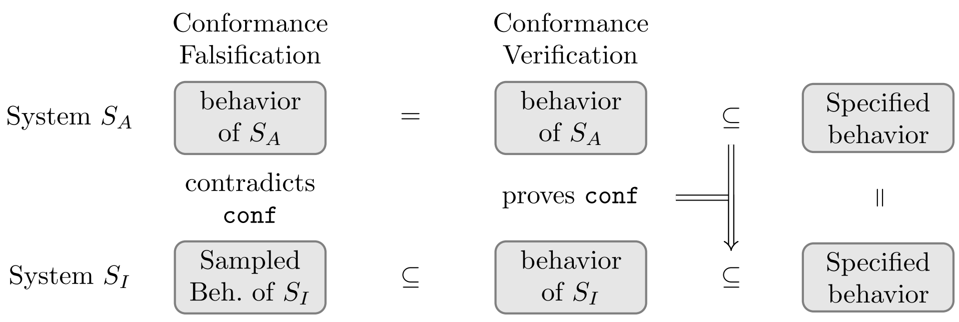

Fig. 3. Conformance Overview.

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 [30]. Conformance should be defined as permissive as possible to relate many systems, yet as strong as necessary to transfer the properties of interest.

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 behaviors of two systems are conformant. This is typically only possible for well-defined white box models or for measurements with further assumptions on the system behavior, because otherwise we cannot prove that we have checked every possible behavior. On the other hand, conformance falsification only needs sampled behavior 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, the confidence that the considered systems are conformant is increased, but does not provide a formal proof. Although conformance falsification is incomplete, conformance is tested in practice with finitely many tests, which are selected to obtain a high confidence that most (relevant) behaviors have 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. This implies selecting 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 (e.g., freedom of collisions of an autonomous vehicle). Linear-time properties specify desired discrete traces of a discrete system [25, Section 3.2.3]. With Linear Temporal Logic (LTL) [25, Section 5], a set of desired traces can be specified by combining propositional formulas on states with temporal operators, such as *always* and *eventually*. Computational Tree Logic (CTL) [25] 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 [25, Section 6.1]. For timed systems, there are timed versions of LTL and CTL, called timed LTL (TLTL) and timed CTL (TCTL) [25, 113, 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 continuous time [16]. Furthermore, MTL has been extended to Signal

Temporal Logic (STL) [90], which maps propositions on continuous states to Boolean traces and thus can be used to define temporal properties for hybrid systems.

One additional topic – highlighting a big difference between discrete and hybrid systems – is reachability analysis, which computes reachable states given a dynamical system, a set of initial states, and inputs. Reachability analysis is decidable for discrete and finite systems [25] but not for hybrid systems [15]. While reachable sets can be exactly computed for discrete systems [33], they cannot be computed exactly for most system classes involving continuous dynamics, and therefore over-approximations are computed [104]. For linear continuous dynamics in particular, relatively high-dimensional state spaces can be efficiently computed [12, 27, 32, 61]. However, reachability analysis remains a difficult (and undecidable) problem for non-linear and hybrid dynamics [13, 14, 26, 39, 44, 56], which affects the application of conformance testing approaches.

3 CONFORMANCE RELATIONS

The main idea of conformance relations is that the inner structure and the state of systems are not relevant, as long as the behavior on the output is similar. In this section, we give an overview of existing conformance relations and their test procedures. Hybrid systems are a superclass of both discrete and timed systems. Thus, all conformance relations for discrete and timed systems can also be applied to hybrid systems in principle. For instance, conformance relations for discrete systems are usually defined on transition systems and can also be applied to infinite transition systems, which can be used as the modeling formalism for hybrid systems [135]. However, discrete conformance relations are meant to be used for discrete systems and might not capture the continuous part of hybrid systems well (cf. Section 3.1). Therefore, additional conformance relations for hybrid systems were proposed, which are based on existing discrete conformance relations. As discussed in the introduction, this survey focuses on such conformance relations. As systems can also be abstracted to discrete systems and compared using discrete conformance relations, we also include references and some explanation to the underlying discrete and timed relations. This helps to understand the foundation and origin of the hybrid conformance relations.

We classify conformance relations as presented in Figure 4: *Simulation* [124] is a conformance relation that relates states; *Trace conformance* [43] only relates output traces and not states; *Reachset conformance* [116] abstracts the set of traces and only checks these abstractions; *Approximate simulation* [59] and *approximate trace conformance* [65] 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 four parts: (i) definition, (ii) properties and transference, (iii) conformance verification and falsification, and (iv) underlying discrete and timed relations. After presenting the different conformance relations, we compare them in Section 4. While simulation and trace conformance are hybrid adaptions of discrete conformance relations, it is noteworthy that the other relations differ much more from discrete conformance relations.

3.1 Trace conformance relations

The basic idea of trace conformance relations is that two systems are conformant if all traces of the first system are also traces of the second system. The concept was initially developed for discrete systems (see the paragraph about discrete conformance below). Based on these

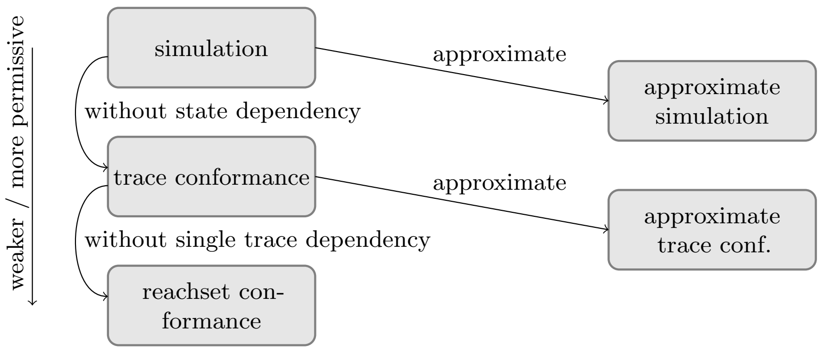

Fig. 4. Conformance relations overview.

existing discrete conformance relations, Dang [43, 45], and similarly van Osch [135, 136], has defined a similar conformance notion for hybrid systems:

Definition. System S_I is trace conformant to system S_A, which we write as $S_I\,\texttt{conf}_T\,S_A$, if the output traces of one system are included in the set of traces of the other system:

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

Following Dang [43], we call this relation *trace conformance*; 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 labeled transition systems as the system modeling 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 essentially the same notion of conformance for different formalism. Alur et al. [17] use *language inclusion* for hierarchical hybrid systems and Henzinger et al. [18, 72] *refinements* for hierarchical hybrid systems. Lynch et al. [86] introduce *implementation relations* 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 [68] presents trajectory propagation and trajectory lifting. Ikeda et al. [41, 75] present the inclusion principle – which is essentially trace conformance – for linear systems in the context of decentralized control. Mitsch et al. [93] discuss *projective relational refinement* for refactoring and refining hybrid systems given in differential dynamic logic. Following Quesel [112], we call two systems *trace equivalent* if trace conformance holds in both directions. Alur et al. [20] call it *language equivalence*, whereas Pola et al. [107] 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. [20] show that trace equivalence transfers linear

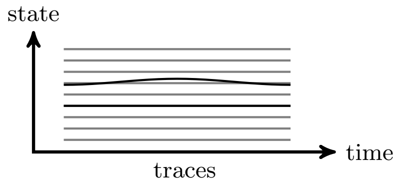

Fig. 5. The gray output traces contain only one of the black output traces. Contrary to the gray traces, the other black output trace is non-constant and thus, the black output traces are not trace conformant to the gray traces.

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 labeled with pass or fail. Paths starting at the root of the tree represent traces of the system. Finite time evolutions of the continuous parts are included as elements of the discrete test tree. A test execution traverses the tree and finally leads to a pass or fail in 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 behaviors. Structural information from the hybrid systems is not leveraged, but naively transformed into hybrid labeled transition systems and test trees.

Dang [43, 45] 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] [43, 46]. 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. [92] 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 a model.

Underlying discrete and timed relations. The underlying discrete version of trace conformance is called *trace inclusion* and assumes what was described above: all sequences of events that can be produced by the first system must be reproducible by the second system [25]. Input events and output events are just considered as events; no distinction between input and output events is made. Trace inclusion transfers LTL properties for systems without terminal states [25, Theorem 3.15].

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

When distinguishing between input events and output events, 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 non-admissible based on the state of the system [111]. Quiescence describes a state of the system in which there is no observable output.

The input-output conformance relation (ioco) [132] can be seen as the counterpart of trace conformance for systems with input-output distinction. Ioco assumes input-enabledness and tackles quiescences with a synthetic/virtual output to make quiescent situations observable. 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. Furthermore, there are other conformance notions for input-output systems, which do not assume input-enabledness; see for instance refinement calculus [24].

Ioco uses a virtual output for quiescent situations, because it cannot measure time. Ideally, there are timeouts that decide whether an output has been quiescent, so that conformance can be decided after the time bound has elapsed. Such timeouts can be defined for timed systems, as time is also an output. This is done by *tioco*, a timed version of ioco [82].

The following conformance relation is defined on hybrid systems; however, it compares the systems on a discrete abstraction. Therefore, we have added this relation to the discrete relations paragraph. The name of the abstraction-based conformance relation is *qualitative reasoning input output conformance* (qrioco) [8, 35]. Qualitative reasoning models 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. [34] propose critera domain coverage, delta coverage, complete delta coverage, state coverage, and transition coverage on the qualitative-reasoning model for test generation. Although qrioco considers hybrid systems, it compares the systems on a discrete abstraction. Thus, evolutions of the continuous states within the equivalence classes are irrelevant for qrioco conformance.

3.2 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

$$S_I \,\texttt{conf}_{\approx}\, S_A \Leftrightarrow \forall u \in inputs(S_A)\ \forall T_{S_I} \in otraces(S_I, u)\ \exists T_{S_A} \in otraces(S_A, u) : d(T_{S_I}, T_{S_A}) \leq \varepsilon.$$

If one uses the metric $d(T_{S_I}, T_{S_A}) = \sup_t d_o(T_{S_I}(t), T_{S_A}(t))$ with a metric d_o on the output space, one obtains the *approximate language inclusion* as defined by Girard and Pappas [65]. However, they only used it for comparison to approximate simulation – contrary to Bian and Abate [31], 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. [28].

We want to emphasize that transference in the sense of Section 2 does not hold for approximate trace conformance relations. Instead, system S_A has to satisfy a robust version

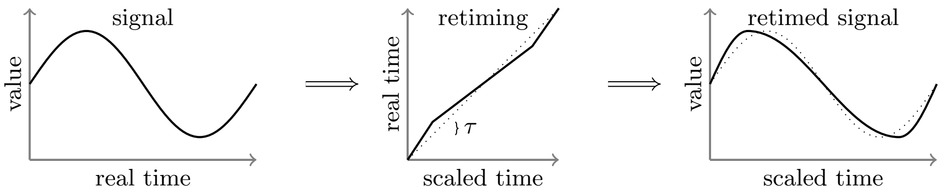

Fig. 6. A retiming function transforms a signal.

$[spec]_{\approx\text{-robust}}$ of the specification to verify that *spec* holds on S_I:

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

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 modeled 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 7. The following approximate relations focus on such systems and allow output deviation and time deviation. In the case of non-continuous jumps of signals in particular, retiming plays an important role. For an easier presentation, we use *retiming functions* [112], which simply scale time (not necessarily continuously).

(τ, ε)*-closeness relation.* Abbas et al. [3] present a conformance relation called (τ, ε)-closeness focusing on deterministic models with continuous outputs. They consider a bounded output deviation ε, and additionally, a bounded time deviation τ as visualized in Figure 6. The time deviation bound τ can be represented by a retiming function r as

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

Thus, 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. [3] define (τ, ε)-closeness for two deterministic systems S_I and S_A as

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

where T_I and T_A are the (single) output traces for a given input and initial state. Note that Abbas et al. [3] define (τ, ε)-closeness for sampled output traces with a finite horizon and thus, their definition is directly applicable to numerically simulated output traces and measured ones. Depending on the application, a slightly different version of (τ, ε)-closeness defined on hybrid time [3] can be used, which requests that the number of discrete state jumps is the same for both output traces. Mohaqeqi and Mousavi [94] extended (τ, ε)-closeness to also incorporate discrete actions. Additionally, Mohaqeqi et al. [96] study the differences of (τ, ε)-closeness and hioco.

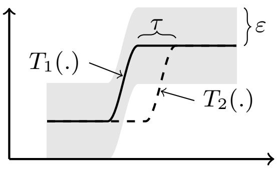

Fig. 7. The output trace $T_2(.)$ is similar to output trace $T_1(.)$, but with a time delay τ. The output distance of $T_1(t)$ and $T_2(t)$ is greater than ε for some time t.

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. [4] 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. [2].

To determine the (τ, ε)-closeness between systems, Abbas et al. [2] 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 [101], or ant colony optimization [21], 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 [95] 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. [23]. Aerts et al. [5] 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). Also, (τ, ϵ)-closeness [29] was used to check a DC-DC converter.

ε-δ-similarity. Quesel [112] introduces ε-δ-similarity, which is (τ, ε)-closeness with $\varepsilon = \delta$ and a preserved time order. Therefore, the retiming r has to satisfy $t_1 < t_2 \Leftrightarrow r(t_1) < r(t_2)$. Transference theorems are proved for this relation: region stability can be transferred, as well as an MTL fragment. Quesel performs conformance checking using quantifier elimination in KeYmaera [105]. Due to the complexity of quantifier elimination, the approach seems to be limited to rather simple models.

ε-Skorokhod conformance. Deshmukh et al. [48] 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 quantifiers (subsuming STL) with ε-Skorokhod conformance. The authors have implemented a stochastic search-based approach maximizing 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 [88, 89] 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 [48].

3.3 (Bi-)simulation relations

The notion of simulation and bisimulation originated from studying discrete structures in theoretical computer science [117]. Tabuada [124] and Frehse [51] 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 [51, 86, 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 strace x_A of S_A starting at s_A such that

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

holds, where $(q_I, x_I(.))$ and $(q_A, x_A(.))$ are parts of the sequences x_I and x_A, respectively. Note that the classical definition of a simulation relation on hybrid automata does not consider inputs. However, by forcing the state traces x_A and x_I to have the same input $u(.)$, inputs can be incorporated. Cuijpers [42] and Prabhakar et al. [108–110] define stronger relations called continuous simulation and uniformly continuous (input-output) simulation, which require the relation between states to be (uniformly) continuous.

If a simulation relation holds in both directions simultaneously, the systems are called bisimilar. Bisimulation relations have been used for linear systems [103, 134], for non-linear systems [125], for switching linear systems [107], and for hybrid systems [37, 124, 133]. The theory of bisimulations for dynamical and hybrid systems was unified by Haghverdi et al. [70] 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. [20]. 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 non-linear systems is presented by Ho and Grasse [69, 74]. However, the classical notion of simulation does not transfer stability properties [42]. Stability transference requires a continuous simulation [42], 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 relations between two models and not between a model and an implementation, e.g., the tool PHAVer [52] can be used to check simulation relations [51, 55]. This is because simulation relates system states, which are typically not possible to obtain from measurements. For linear systems, there exist methods to compute bisimulation relations between given models using a fixed-point characterization [103, 134]. Tanner and Pappas [129] present necessary and 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 [97] present a method to compute simulation relations between non-linear control systems affine in inputs and disturbances. Murthy et al. [99, 100] present the framework BFComp, which computes bisimulation functions using sum-of-squares optimization, δ-decidability over the reals, and counter-example guided search. Yang [139] generalizes the scalar-valued simulation functions to vector-valued simulation functions, which can be used to prove that a simulation relation holds.

Underlying discrete and timed relations. One application of the notions of simulation and bisimulation is to abstract a discrete system to a bisimilar quotient system for verification purposes [25, Section 7.1.1]. "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." [25, Section 7.1]. A detailed study of bisimulation was conducted by Roggenbach and Majster-Cederbaum [117], and for probabilistic bisimulation by Abate [1]. While CTL [25, Section 7.2] properties can be transferred with bisimulation, only a subclass of CTL [25, Section 7.5] properties can be transferred with simulation. Compared to trace conformance relations, simulation demands more similarity between the system to be conformant. This follows from the following implications: (i) Simulation implies trace inclusion, if there are no terminal states [25, Theorem 7.70], and (ii) bisimulation implies trace equivalence [25, Theorem 7.6]. A detailed comparison of bisimulation, simulation, and trace equivalence can be found in [25, Section 7.4.2]. The extension of simulation from discrete systems to timed systems led to timed simulation [130]. Timed simulation has been studied for hybrid systems [53] and resulted in subclasses of hybrid automata which are composable with respect to timed simulation.

3.4 Approximate simulation relations

For systems where the continuous dynamics are the main concern, the exact notions of simulation relations can be too restrictive. However, real-valued metrics can be used to quantify distances. This led to the generalization of simulation relations to approximate simulation relations, used for instance in system biology [98]. 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 0, approximate simulation becomes actual simulation. The approximate nature is used to relate a model or implementation to a simpler model which does not model every detail, and thus has some deviation in the outputs. As a result, one has to change properties on transference as discussed for approximate trace conformance relations in Sec. 3.2.

Definition. Approximate simulation is defined on metric transition systems which are transition systems combined with a metric d on the output space [59]. 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 [58] and co-authors. They defined approximate simulation for continuous systems [65], as well as for hybrid systems [59, 60]. 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 [102]. Tabuada [123] defines (ε, δ)-approximate simulation motivated by stability properties. The (ε, δ)-approximate simulation relation is basically the same as ε-approximate simulation relation, 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 $(s_I, s_A) \in R$ (see Eq. (3)).

The symmetric relation – both systems approximately simulate each other – is called approximate bisimulation. If it holds, output traces are at most ε away of being equivalent. Approximate bisimulations have been defined for constrained linear systems [62, 64], for non-linear dynamical systems [63], and for hybrid systems [59, 60].

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. [77] 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 the scope of this survey. However, we want to shortly mention that there are also definitions of approximate simulation for linear stochastic systems [78] and hybrid stochastic systems [76, 79]. Abate [1] 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 [59, 60], 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 [59, 65] and bisimulation functions [63, 65], 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 [77].

Conformance verification and falsification. Several techniques to compute simulation functions have been developed. These simulation functions can be used to derive simulation relations. The first one reformulates the simulation function search as a linear matrix inequality (LMI) problem [62, 64, 79, 87]. The solution of the LMI generates a simulation function. Unfortunately, these methods are restricted to linear systems only. For non-linear systems, bisimulation functions can be computed by solving a sum-of-squares (SOS) problem [63]. 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 to generate a simulation function. Therefore, an interactive approach is not possible, and one gains no insight into the system if no simulation function is generated. Yan et al. [138] have proposed a method to compute

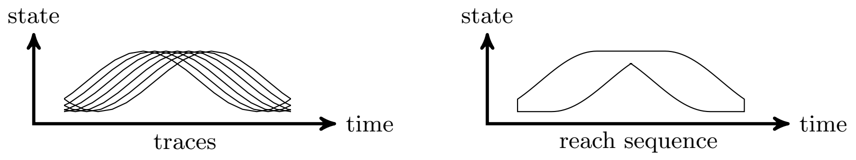

Fig. 8. The reach sequence of a set of traces as published by Roehm et al. [116].

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 [57]. There are also methods to construct approximate simulating models for a given model [67, 106] and for control purposes [66].

Underlying discrete and timed relations. Discrete systems typically do not provide a useful metric as is the case for hybrid systems. We are not aware of any approximate notion of simulation for discrete systems. This is different for timed systems, as we can compute the time difference between events. Approximate timed simulation [71] allows a bounded deviation between the times at which events happen in both systems.

3.5 Reachset conformance relation

So far, we have presented conformance relations that relate traces in an exact or approximate fashion. The following relation does something else: it compares abstractions on 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 behavior from the ideal behavior should also be safe.

Definition. Formally, the reachset conformance between S_I and S_A is

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

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 abstracting time version *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. [85] introduce differential refinement logic d$\mathcal{RL}$, which syntactically extends differential dynamic logic with a refinement operator on hybrid systems. If the conformance relation is not mixed with the system models, d$\mathcal{RL}$ is identical to weak reachset conformance. However, since mixing the refinement operator with differential dynamical logic is possible, other conformance relations can 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 transferring 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 [11], as well as measured data from real systems [116], to underapproximate the reachable sets. Thus, 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 [43]. Reachset conformance has already been used to check conformance of a model of walking humans [84].

4 COMPARISON

In the previous 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 examples to better explain the relations.

4.1 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 relation should be preferred over approximate simulation relation, when possible, unless both systems deviate from each other to some extent. In that case, approximate simulation can be used to relate these systems. If computational tree logic (TCTL) properties are of interest instead of linear-time properties, bisimulation has to be used instead of simulation. Note that formal conformance verification takes a considerable effort and thus is restricted to small system models on relatively high abstraction levels.

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 required 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 [10], SpaceEx [54], or Flow* [38].

Table 1. Properties which guide the conformance relation selection process between systems S_I and S_A. Abbreviations are used for white box (□), black box (■), formal verification (FV), and testing (T).

Conf. Relation	S_I	S_A	Focus	Transference of
Bisimulation	□	□	FV	safety, TCTL, TLTL, MTL, STL
Simulation	□	□	FV	safety, TCTL (partially), TLTL, MTL, STL
Approx. Simulation	□	□	FV	similar to simulation[2]
Trace Conformance	■	■	T	safety, TLTL, MTL, STL
Approx. Trace Conf.	■	■	T	similar to trace conformance[2]
Reachset Conf.	■	□	T	safety, RTL

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 that approximate conformance relations do not transfer the exact temporal property but a slightly changed one [4]. The approach can also be used for black-box models and systems with only their inputs and outputs being accessible.

4.2 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 [65, 86, 107, 124, 134]. 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. All three models have two continuous state dimensions x, and y. The output is the projection to x and no input is considered.

- The model M1 has two output traces: a sine and a cosine function for initial states $x = 0$ and $y = 1$.
- The model M2 selects, at time $\pi/2$, which function, sine or cosine, it switches to after 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 each half period represented by the transition label $x = 0$ [91]. 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 is not the case for M3.

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

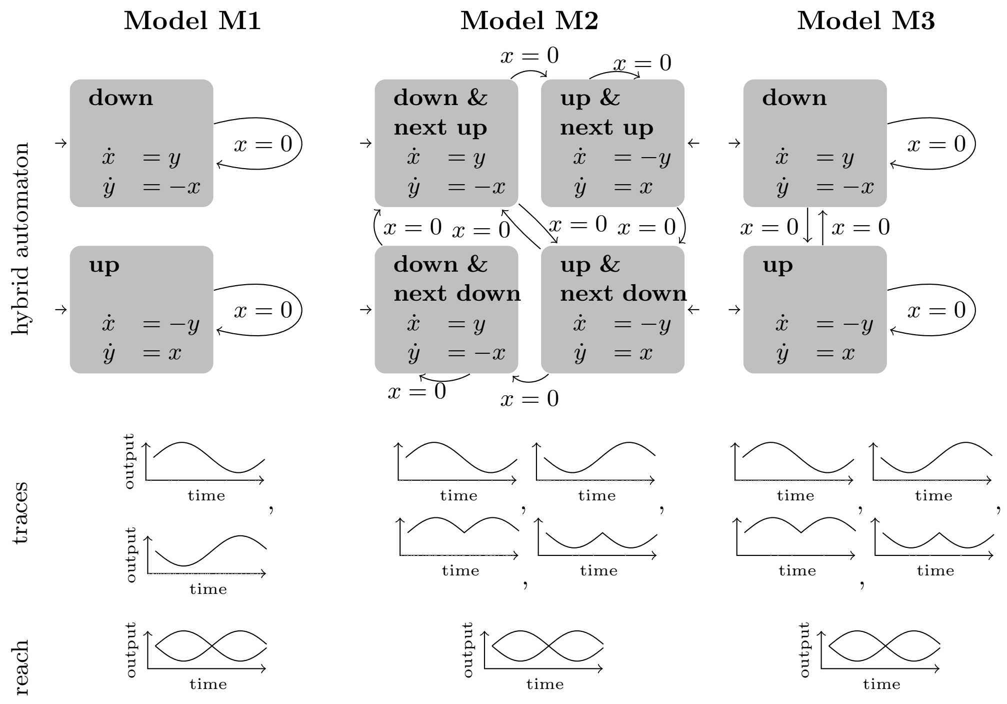

Fig. 9. Three example models and their reachsets and output traces.

Is the model in the column reachset conf. to the model in the row?

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

Is the model in the column trace conf. to the model in the row?

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

Is the model in the column simulated by the model in the row?

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

Fig. 10. Comparison of conformance for different models.

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 on the difference between simulation and trace conformance. Roehm et al. [116] do the same for reachset conformance and trace conformance. In these examples, the source of non-determinism is the continuous side; this shows that the differences between the relations do persist if the source of non-determinism does not come from the discrete side.

4.3 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,

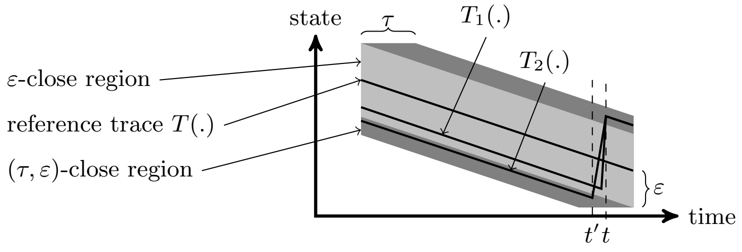

Fig. 11. Visualization of reachsets, which represent reference-trace-close traces for approximate language inclusion and (τ, ε)-closeness, as well as two sample traces.

we discuss the differences through an example: 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 a trace $T(.)$ can be represented by tubes. These tubes can be represented as reachset sequences, e.g.,

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

for an one-dimensional output trace T', and the conformance 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 is therefore less restrictive. Conformance relations, where the output traces are ε-close to $T(.)$ and 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 preventing 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

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

4.4 Approximate trace relations vs. approximate simulation

For the approximate versions of trace conformance and simulation, the differences are similar to those of the exact versions. Girard et al. [59, 60, 63] 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 [3].

Our short comparison of conformance relations neglects the fact that the relations are defined on different modeling 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 modeling 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 [81] 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.

5 INPUT SELECTION FOR CONFORMANCE TESTING

Lee [83] 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 (also called test case generation). 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 behavior or to give a high confidence that we have tested conformance sufficiently, e.g., using a test-end criterion.

Ideally, the selected inputs come with a formal proof that they sufficiently show conformance between the system. For discrete systems, there exists a variety of methods to generate inputs [36]. There are methods to construct a finite set of test inputs, which are sufficient to show that the system under test is conformant, as long as the number of finite states can be bounded [40]. For hybrid systems, this is much more difficult as reachability is already undecidable compared to discrete systems.

Inputs should be selected such that we find non-conformant behavior 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 the 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. [6]).

Coverage. A notion of coverage can support the input selection towards diversification of input selection for testing. 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 so that discrete coverage measures can be used. 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 [43]. 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.

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 (cf. [45]). A possible approach is to approximatively cover the reachable state space of S_A with a finite number of

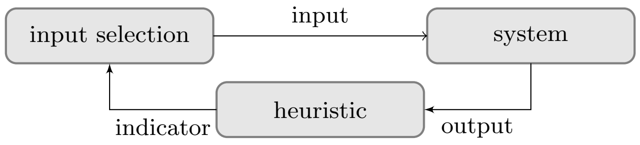

Fig. 12. Generic incremental input selection approach.

tests [22, 49, 50, 80]. This ensures that the inputs are selected such that sufficiently different behaviors of the system are shown. Brandl et al. [34] measure coverage based on the coverage of a discrete abstraction of the continuous state space, called qualitative reasoning models. For discussion of coverage for timed systems, see [82].

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 [43, 47] 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 [43]. 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 behavior is the use of mutant-based methods [7, 9, 114].

Covering the reachable state space of S_A can be seen as the continuous counterpart of statement coverage. 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 preferred 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 [21, 101] and has been transfered to work for falsifying the conformance relations $(\tau - \varepsilon)$-closeness [3] and ε-Skorokhod conformance [48]. 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 to find non-conformant behavior. If a metric is available, this approach can be used for all other present notions of conformance as well. For reachset conformance in particular, 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 behavior, relevant behavior should be exposed. This can be achieved by combining coverage-based input selection with optimization-based 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 a different type of behavior was exposed.

Since we can only run a finite number of test cases, we need a test-end criterion at which point we can confidently stop conformance testing. In general, this requires assumptions on the systems under test and thus, this is application-specific. Dang uses disparity of consecutive tests as a termination criterion [43]. 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. [43, Section 1.10]).

6 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 which 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 them. Open questions include test-coverage criteria, guarantees with respect to assumptions on system behavior, and test-input-generation algorithms. From an industrial standpoint, we believe that the applicability of the theoretical conformance relations hinges on providing tailored means to transport academic results into practice by systematic testing.

REFERENCES

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

[2] H. Abbas and G. Fainekos. 2015. *Towards composition of conformant systems.* Technical Report. https://arxiv.org/abs/1511.05273

[3] H. Abbas, B. Hoxha, G. E. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. 2014. Conformance Testing as Falsification for Cyber-Physical Systems. *CoRR* abs/1401.5200 (2014). http://arxiv.org/abs/1401.5200

[4] H. Abbas, H. D. Mittelmann, and G. E. Fainekos. 2014. Formal property verification in a conformance testing framework. In *Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE.* 155–164. https://doi.org/10.1109/MEMCOD.2014.6961854

[5] A. Aerts, M. R. Mousavi, and M. A. Reniers. 2015. A Tool Prototype for Model-Based Testing of Cyber-Physical Systems. In *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium.* 563–572. https://doi.org/10.1007/978-3-319-25150-9_32

[6] A. Aerts, M. Reniers, and M.R. Mousavi. 2017. Chapter 19 - Model-Based Testing of Cyber-Physical Systems. In *Cyber-Physical Systems.* Academic Press, 287 – 304. https://doi.org/10.1016/B978-0-12-803801-7.00019-5

[7] B. K. Aichernig, H. Brandl, E. Jöbstl, and W. Krenn. 2009. Model-Based Mutation Testing of Hybrid Systems. In *Formal Methods for Components and Objects - 8th International Symposium, FMCO.* 228–249. https://doi.org/10.1007/978-3-642-17071-3_12

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

[9] B. K. Aichernig, F. Lorber, and D. Nickovic. 2013. Time for Mutants - Model-Based Mutation Testing with Timed Automata. In *Tests and Proofs - 7th International Conference, TAP.* 20–38. https://doi.org/10.1007/978-3-642-38916-0_2

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

[11] M. Althoff and J. M. Dolan. 2012. Reachability computation of low-order models for the safety verification of high-order road vehicle models. In *American Control Conference, ACC.* 3559–3566. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=6314777

[12] M. Althoff and G. Frehse. 2016. Combining Zonotopes and Support Functions for Efficient Reachability Analysis of Linear Systems. In *Proc. of the 55th IEEE Conference on Decision and Control*. 7439–7446.

[13] M. Althoff and B. H. Krogh. 2012. Avoiding Geometric Intersection Operations in Reachability Analysis of Hybrid Systems. In *Hybrid Systems: Computation and Control*. 45–54.

[14] M. Althoff and B. H. Krogh. 2014. Reachability Analysis of Nonlinear Differential-Algebraic Systems. *IEEE Trans. Automat. Control* 59, 2 (2014), 371–383.

[15] R. Alur, C. Courcoubetis, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. 1994. The algorithmic analysis of hybrid systems. In *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*. Springer, 329–351.

[16] R. Alur, T. Feder, and T. A. Henzinger. 1996. The Benefits of Relaxing Punctuality. *J. ACM* 43, 1 (1996), 116–146. https://doi.org/10.1145/227595.227602

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

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

[19] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. 1998. Alternating Refinement Relations. In *Concurrency Theory, 9th International Conference CONCUR*. 163–178. https://doi.org/10.1007/BFb0055622

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

[21] Y. S. R. Annapureddy and G. E. Fainekos. 2010. Ant Colonies for Temporal Logic Falsification of Hybrid Systems. In *Proc. of the 36th Annual Conference of IEEE Industrial Electronics*. 91–96.

[22] D. Araiza-Illan, D. Western, A. Pipe, and K. Eder. 2016. Systematic and Realistic Testing in Simulation of Control Code for Robots in Collaborative Human-Robot Interactions. In *Towards Autonomous Robotic Systems: 17th Annual Conference*. 20–32.

[23] H. Araujo, G. Carvalho, A. Sampaio, M. R. Mousavi, and M. Taromirad. 2017. A Process for Sound Conformance Testing of Cyber-Physical Systems. In *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 46–50. https://doi.org/10.1109/ICSTW.2017.15

[24] R.-J. Back and J. von Wright. 1998. *Refinement Calculus - A Systematic Introduction*. Springer. https://doi.org/10.1007/978-1-4612-1674-2

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

[26] S. Bak and P. S. Duggirala. 2017. HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems. In *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*. 173–178.

[27] S. Bak and P. S. Duggirala. 2017. Simulation-Equivalent Reachability of Large Linear Systems with Inputs. In *Proceedings of the 29th International Conference on Computer Aided Verification*. Springer.

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

[29] O. Beg, H. Abbas, T. T. Johnson, and A. Davoudi. 2017. Model Validation of PWM DC-DC Converters. *IEEE Trans. Industrial Electronics* 64, 9 (2017), 7049–7059. https://doi.org/10.1109/TIE.2017.2688961

[30] S. Bensalem, A. Bouajjani, C. Loiseaux, and J. Sifakis. 1992. Property Preserving Simulations. In *Computer Aided Verification, Fourth International Workshop, CAV*. 260–273. https://doi.org/10.1007/3-540-56496-9_21

[31] G. Bian and A. Abate. 2017. On the Relationship between Bisimulation and Trace Equivalence in an Approximate Probabilistic Context. In *International Conference on Foundations of Software Science and Computation Structures*. Springer, 321–337.

[32] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling. 2018. Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC*. 41–50. https://doi.org/10.1145/3178126.3178128

[33] A. Bouajjani, J. Esparza, and O. Maler. 1997. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. of the 8th International Conference on Concurrency Theory*. 135–150.

[34] H. Brandl, G. Fraser, and F. Wotawa. 2008. Coverage-based Testing Using Qualitative Reasoning Models. In *Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE)*. 393–398.

[35] H. Brandl, M. Weiglhofer, and B. K. Aichernig. 2010. Automated conformance verification of hybrid systems. In *10th International Conference on Quality Software (QSIC)*. 3–12.
[36] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner (Eds.). 2005. *Model-Based Testing of Reactive Systems, Advanced Lectures*. Lecture Notes in Computer Science, Vol. 3472. Springer. https://doi.org/10.1007/b137241
[37] M. L. Bujorianu, J. Lygeros, and Marius C Bujorianu. 2005. Bisimulation for general stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 198–214.
[38] X. Chen, E. Ábrahám, and S. Sankaranarayanan. 2013. Flow*: An Analyzer for Non-Linear Hybrid Systems. In *Proc. of Computer-Aided Verification (LNCS 8044)*. Springer, 258–263.
[39] X. Chen, M. Althoff, and F. Immler. 2017. ARCH-COMP17 Category Report: Continuous Systems with Nonlinear Dynamics. In *Proc. of the 4th International Workshop on Applied Verification for Continuous and Hybrid Systems*. 160–169.
[40] T. S. Chow. 1978. Testing software design modeled by finite-state machines. *IEEE transactions on software engineering* 3 (1978), 178–187.
[41] D. Chu and D. D. Siljak. 2005. A canonical form for the inclusion principle of dynamic systems. *SIAM Journal on Control and Optimization* 44, 3 (2005), 969–990.
[42] P. J. L. Cuijpers. 2007. On bicontinuous bisimulation and the preservation of stability. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 676–679.
[43] T. Dang. 2011. Model-based Testing of Hybrid Systems. In *Model-Based Testing for Embedded Systems*. CRC Press, Inc., Chapter 14, 383–424.
[44] T. Dang, O. Maler, and R. Testylier. 2010. Accurate Hybridization of Nonlinear Systems. In *Hybrid Systems: Computation and Control*. 11–19.
[45] T. Dang and T. Nahhal. 2009. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design* 34, 2 (2009), 183–213.
[46] T. Dang and T. Nahhal. Nov 2007. *Model-based testing of hybrid systems*. Technical Report. Verimag, IMAG.
[47] T. Dang and N. Shalev. 2014. Test Coverage Estimation Using Threshold Accepting. In *Automated Technology for Verification and Analysis*. Vol. 8837. Springer International Publishing, 115–128. https://doi.org/10.1007/978-3-319-11936-6_9
[48] J. V. Deshmukh, R. Majumdar, and V. S. Prabhu. 2015. Quantifying Conformance Using the Skorokhod Metric. In *Computer Aided Verification - 27th International Conference, CAV*. 234–250. https://doi.org/10.1007/978-3-319-21668-3_14
[49] A. Donzé. 2007. *Trajectory-Based Verification and Controller Synthesis for Continuous and Hybrid Systems*. Ph.D. Dissertation. University Joseph Fourier.
[50] A. Donzé. 2010. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification, 22nd International Conference, CAV*. 167–170. https://doi.org/10.1007/978-3-642-14295-6_17
[51] G. Frehse. 2005. *Compositional Verification of Hybrid Systems Using Simulation Relations*. Ph.D. Dissertation. Radboud Universiteit Nijmegen.
[52] G. Frehse. 2005. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. In *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC*. 258–273. https://doi.org/10.1007/978-3-540-31954-2_17
[53] G. Frehse. 2006. On Timed Simulation Relations for Hybrid Systems and Compositionality. In *Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS*. 200–214. https://doi.org/10.1007/11867340_15
[54] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Computer Aided Verification - 23rd International Conference, CAV*. 379–395. https://doi.org/10.1007/978-3-642-22110-1_30
[55] G. Frehse, Z. Han, and B. Krogh. 2004. Assume-guarantee reasoning for hybrid I/O-automata by over-approximation of continuous interaction. In *43rd IEEE Conference on Decision and Control*, Vol. 1. 479–484.
[56] G. Frehse and R. Ray. 2012. Flowpipe-Guard Intersection for Reachability Computations with Support Functions. In *Proc. of Analysis and Design of Hybrid Systems*. 94–101.
[57] A. Girard. 2013. A composition theorem for bisimulation functions. *arXiv preprint arXiv:1304.5153* (2013).

[58] A. Girard. 2013. *Computational Approaches to Analysis and Control of Hybrid Systems*. https://tel.archives-ouvertes.fr/tel-00908913 Habilitation.
[59] A. Girard, A. A. Julius, and G. J. Pappas. 2006. Approximate simulation relations for hybrid systems. *IFAC Proceedings Volumes* 39, 5 (2006), 106–111.
[60] A. Girard, A. A. Julius, and G. J. Pappas. 2008. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems* 18, 2 (2008), 163–179.
[61] A. Girard and C. Le Guernic. 2008. Efficient Reachability Analysis for Linear Systems using Support Functions. In *Proc. of the 17th IFAC World Congress*. 8966–8971.
[62] A. Girard and G. J. Pappas. 2005. Approximate bisimulations for constrained linear systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 4700–4705.
[63] A. Girard and G. J. Pappas. 2005. Approximate Bisimulations for Nonlinear Dynamical Systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*. 684–689. https://doi.org/10.1109/CDC.2005.1582235
[64] A. Girard and G. J. Pappas. 2007. Approximate bisimulation relations for constrained linear systems. *Automatica* 43, 8 (2007), 1307 – 1317. https://doi.org/10.1016/j.automatica.2007.01.019
[65] A. Girard and G. J. Pappas. 2007. Approximation Metrics for Discrete and Continuous Systems. *Automatic Control, IEEE Transactions on* 52, 5 (May 2007), 782–798. https://doi.org/10.1109/TAC.2007.895849
[66] A. Girard and G. J. Pappas. 2009. Hierarchical control system design using approximate simulation. *Automatica* 45, 2 (2009), 566–571.
[67] A. Girard, G. Pola, and P. Tabuada. 2010. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Trans. Automat. Control* 55, 1 (2010), 116–126.
[68] K. A. Grasse. 2007. Simulation and Bisimulation of Nonlinear Control Systems with Admissible Classes of Inputs and Disturbances. *SIAM J. Control Optim.* 46, 2 (April 2007), 562–584. https://doi.org/10.1137/050638072
[69] K. A. Grasse and N. Ho. 2015. Simulation relations and controllability properties of linear and nonlinear control systems. *SIAM Journal on Control and Optimization* 53, 3 (2015), 1346–1374.
[70] E. Haghverdi, P. Tabuada, and G. J. Pappas. 2005. Bisimulation relations for dynamical, control, and hybrid systems. *Theor. Comput. Sci.* 342, 2-3 (2005), 229–261. https://doi.org/10.1016/j.tcs.2005.03.045
[71] T. A. Henzinger, R. Majumdar, and V. S. Prabhu. 2005. Quantifying Similarities Between Timed Systems. In *Third International Conference on Formal Modeling and Analysis of Timed Systems FORMATS*. 226–241. https://doi.org/10.1007/11603009_18
[72] T. A. Henzinger, M. Minea, and V. Prabhu. 2001. Assume-guarantee reasoning for hierarchical hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 275–290.
[73] 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. 2009. Using formal specifications to support testing. *ACM Comput. Surv.* 41, 2 (2009), 9:1–9:76. https://doi.org/10.1145/1459352.1459354
[74] N. Ho. 2015. *Controllability of Linear and Nonlinear Control Systems Related Through Simulation Relations*. Ph.D. Dissertation. UNIVERSITY OF OKLAHOMA.
[75] M. Ikeda, D. D. Siljak, and D. E. White. 1982. An Inclusion Principle for Dynamic Systems. In *1982 American Control Conference*. 884–892. https://doi.org/10.23919/ACC.1982.4787983
[76] A. A. Julius. 2006. Approximate Abstraction of Stochastic Hybrid Automata. In *9th International Workshop on Hybrid Systems: Computation and Control HSCC (Lecture Notes in Computer Science)*, João P. Hespanha and Ashish Tiwari (Eds.), Vol. 3927. Springer, 318–332. https://doi.org/10.1007/11730637_25
[77] A. A. Julius, A. D'Innocenzo, M. D. Di Benedetto, and G. J. Pappas. 2009. Approximate equivalence and synchronization of metric transition systems. *Systems & Control Letters* 58, 2 (2009), 94–101. https://doi.org/10.1016/j.sysconle.2008.09.001
[78] A. A. Julius, A. Girard, and G. J. Pappas. 2006. Approximate bisimulation for a class of stochastic hybrid systems. In *American Control Conference*. IEEE, 6–pp.
[79] A. A. Julius and G. J. Pappas. 2009. Approximations of stochastic hybrid systems. *IEEE Trans. Automat. Control* 54, 6 (2009), 1193–1203.
[80] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. 2003. On Systematic Simulation of Open Continuous Systems. In *Hybrid Systems: Computation and Control (LNCS 2623)*. Springer, 283–297.

[81] N. Khakpour and M. R. Mousavi. 2015. Notions of Conformance Testing for Cyber-Physical Systems: Overview and Roadmap (Invited Paper). In *26th International Conference on Concurrency Theory (CONCUR)*, Vol. 42. 18–40. https://doi.org/10.4230/LIPIcs.CONCUR.2015.18

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

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

[84] S. B. Liu, H. Roehm, C. Heinzemann, I. Lütkebohle, J. Oehlerking, and M. Althoff. 2017. Provably safe motion of mobile robots in human environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*. 1351–1357. https://doi.org/10.1109/IROS.2017.8202313

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

[86] N. A. Lynch, R. Segala, and F. W. Vaandrager. 2001. Hybrid I/O Automata Revisited. In *4th International Workshop on Hybrid Systems: Computation and Control HSCC*. 403–417. https://doi.org/10.1007/3-540-45351-2_33

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

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

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

[90] O. Maler and D. Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Proceedings of the 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*. 152–166. https://doi.org/10.1007/978-3-540-30206-3_12

[91] I. M. Mitchell. 2007. Comparing Forward and Backward Reachability as Tools for Safety Analysis. In *10th International Workshop on Hybrid Systems: Computation and Control HSCC*. 428–443. https://doi.org/10.1007/978-3-540-71493-4_34

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

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

[94] M. Mohaqeqi and M.R. Mousavi. 2016. 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)*.

[95] M. Mohaqeqi and M. R. Mousavi. 2016. Sound Test-Suites for Cyber-Physical Systems. In *10th International Symposium on Theoretical Aspects of Software Engineering TASE*. 42–48. https://doi.org/10.1109/TASE.2016.33

[96] M. Mohaqeqi, M. R. Mousavi, and W. Taha. 2014. Conformance Testing of Cyber-Physical Systems: A Comparative Study. *ECEASST* 70 (2014). http://journal.ub.tu-berlin.de/eceasst/article/view/982

[97] L. Munteanu and K. A. Grasse. 2015. Constructing simulation relations for IDO systems affine in inputs and disturbances. *Mathematics of Control, Signals, and Systems* 27, 3 (2015), 317–346. https://doi.org/10.1007/s00498-015-0142-5

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

[99] A. Murthy, Md A. Islam, S. A. Smolka, and R. Grosu. 2015. 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*. ACM, 78–87.

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

[101] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivančić, A.Gupta, and G. J. Pappas. 2010. Monte-Carlo Techniques for Falsification of Temporal Properties of Non-Linear Hybrid Systems. In *Hybrid Systems: Computation and Control*. 211–220.

[102] H. Pan, M. Zhang, and Y. Chen. 2011. Approximate Simulation for Metric Hybrid Input/Output Automata. In *5th International Conference on Secure Software Integration & Reliability Improvement Companion (SSIRI-C)*. IEEE, 53–59.

[103] G. J. Pappas. 2003. Bisimilar linear systems. *Automatica* 39, 12 (2003), 2035–2047. https://doi.org/10.1016/j.automatica.2003.07.003

[104] A. Platzer and E. M. Clarke. 2007. The Image Computation Problem in Hybrid Systems Model Checking. In *Hybrid Systems: Computation and Control (LNCS 4416)*. Springer, 473–486.

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

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

[107] G. Pola, A. J. van der Schaft, and M. D. Di Benedetto. 2004. Bisimulation theory for switching linear systems. *43rd IEEE Conference on Decision and Control (CDC)* 2 (Dec 2004), 1406–1411 Vol.2. https://doi.org/10.1109/CDC.2004.1430240

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

[109] P. Prabhakar, G. Dullerud, and M. Viswanathan. 2015. Stability Preserving Simulations and Bisimulations for Hybrid Systems. *IEEE Trans. Automat. Control* 60, 12 (2015), 3210–3225.

[110] P. Prabhakar and J. Liu. 2016. Bisimulations for input-output stability of hybrid systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. 5515–5520. https://doi.org/10.1109/CDC.2016.7799116

[111] V. Preoteasa and S. Tripakis. 2016. Towards Compositional Feedback in Non-Deterministic and Non-Input-Receptive Systems. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science LICS*. 768–777. https://doi.org/10.1145/2933575.2934503

[112] J.-D. Quesel. 2013. *Similarity, Logic, and Games: Bridging Modeling Layers of Hybrid Systems*. Ph.D. Dissertation.

[113] J.-F. Raskin. 1999. *Logics, automata and classical theories for deciding real time*. Ph.D. Dissertation. Facultés universitaires Notre-Dame de la Paix, Namur.

[114] H. Roehm, T. Heinz, and E. C. Mayer. 2017. STLInspector: STL Validation with Guarantees. In *Computer Aided Verification - 29th International Conference, CAV*. 225–232. https://doi.org/10.1007/978-3-319-63387-9_11

[115] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. 2016. STL Model Checking of Continuous and Hybrid Systems. In *Automated Technology for Verification and Analysis - 14th International Symposium, ATVA*. 412–427. https://doi.org/10.1007/978-3-319-46520-3_26

[116] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. 2016. Reachset Conformance Testing of Hybrid Automata. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control HSCC*. 277–286. https://doi.org/10.1145/2883817.2883828

[117] M. Roggenbach and M. Majster-Cederbaum. 2000. Towards a unified view of bisimulation: a comparative study. *Theoretical Computer Science* 238, 1 (2000), 81 – 130. https://doi.org/10.1016/S0304-3975(99)00303-5

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

[119] J. Schmaltz and J. Tretmans. 2008. On Conformance Testing for Timed Systems. In *6th International Conference on Formal Modeling and Analysis of Timed Systems FORMATS*. 250–264. https://doi.org/10.1007/978-3-540-85778-5_18

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

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

[122] T. Strathmann and J. Oehlerking. 2015. Experience Report: Verifying Properties of an Electro-Mechanical Braking System. *ARCH* (2015).

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

[124] P. Tabuada. 2009. *Verification and Control of Hybrid Systems - A Symbolic Approach*. Springer. http://www.springer.com/mathematics/applications/book/978-1-4419-0223-8

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

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

[127] P. Tabuada, G. J. Pappas, and P. Lima. 2004. Compositional Abstractions of Hybrid Control Systems. *Discrete Event Dynamic Systems* 14, 2 (2004), 203–238. https://doi.org/10.1023/B:DISC.0000018571.14789.24

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

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

[130] S. Tasiran. 1998. *Compositional and hierarchical techniques for the formal verification of real-time systems*. Ph.D. Dissertation. University of California at Berkeley.

[131] J. Tretmans. 1992. *A formal approach to conformance testing*. Ph.D. Dissertation. Universiteit Twente.

[132] J. Tretmans. 1999. Testing Concurrent Systems: A Formal Approach. In *10th International Conference on Concurrency Theory CONCUR (Lecture Notes in Computer Science)*, Jos C. M. Baeten and Sjouke Mauw (Eds.), Vol. 1664. Springer, 46–65. https://doi.org/10.1007/3-540-48320-9_6

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

[134] A. van der Schaft. 2004. Equivalence of dynamical systems by bisimulation. *IEEE Trans. Automat. Contr.* 49, 12 (2004), 2160–2172. https://doi.org/10.1109/TAC.2004.838497

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

[136] M. van Osch. 2009. *Automated model-based testing of hybrid systems*. Ph.D. Dissertation. Eindhoven University of Technology.

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

[138] G. Yan, L. Jiao, Y. Li, S. Wang, and N. Zhan. 2016. Approximate bisimulation and discretization of Hybrid CSP. In *21st International Symposium Formal Methods FM*. Springer, 702–720.

[139] K. Yang and H. Ji. 2017. Hierarchical analysis of large-scale control systems via vector simulation function. *Systems & Control Letters* 102 (2017), 74 – 80. https://doi.org/10.1016/j.sysconle.2017.01.010