Formal property verification in a conformance testing framework

Houssam Abbas
School of Electrical, Energy and Computer Engineering
Arizona State University
Tempe, AZ, U.S.A.
Email: hyabbas@asu.edu

Hans Mittelmann
School of Mathematical and Statistical Sciences
Arizona State University
Tempe, AZ, U.S.A.
Email: mittelmann@asu.edu

Georgios Fainekos
School of Computing, Informatics and Decision Systems
Arizona State University
Tempe, AZ, U.S.A.
Email: fainekos@asu.edu

***Abstract*—In model-based design of cyber-physical systems, such as switched mixed-signal circuits or software-controlled physical systems, it is common to develop a sequence of system models of different fidelity and complexity, each appropriate for a particular design or verification task. In such a sequence, one model is often derived from the other by a process of simplification or implementation. E.g. a Simulink model might be implemented on an embedded processor via automatic code generation. Three questions naturally present themselves: how do we quantify closeness between the two systems? How can we measure such closeness? If the original system satisfies some formal property, can we automatically infer what properties are then satisfied by the derived model? This paper addresses all three questions: we quantify the closeness between original and derived model via a distance measure between their outputs. We then propose two computational methods for approximating this closeness measure. Finally, we derive syntactical re-writing rules which, when applied to a Metric Temporal Logic specification satisfied by the original model, produce a formula satisfied by the derived model. We demonstrate the soundness of the theory with several experiments.**

I. Introduction

In the last decade, systems which use embedded software to control continuously changing physical phenomena have come to be seen as 'Cyber-Physical Systems' (CPS), a category of systems whose main characteristic is the interaction of continuous and discrete dynamics, possibly with communication between remote components. This comes as a recognition that verifying hardware separately from software, or the physical separately from the cyber, is becoming less satisfactory as the interactions between the two become richer and more complicated, and as the design process needs to guarantee extra-functional requirements [7], [31]. For example, the 2014 Toyota recall of 700,000 Prius vehicles was partly blamed on the interaction between the controller software and the transistors of the control board [36]. In a typical Model-Based Design (MBD) process of CPS (see Fig. 1), a series of models and implementations are iteratively developed such that the end product satisfies a set of formal functional requirements Φ [11]. Ideally, the initial (simpler) model M_S developed should be amenable to formal synthesis and verification methods (cycle 1 in Fig. 1) through tools like [17], [34]. Then, the fidelity of the models is increased by modeling more complex physical phenomena ignored initially, by taking into account non-functional requirements like power consumption, and by modeling inaccuracies introduced by the real-time computational platforms such as look-up-tables, time delays, clock drift, a different computation precision, etc. This yields successively more complex models M_C (cycle 2 in Fig. 1). Afterwards, the model M_C is implemented on a computational platform to yield the prototype S_i; S_i is then manually modified and calibrated into a final deployment system S_d. Finally, if the system is deployed over a communication network, the network will introduce a whole range of issues related to random transmission quality and delayed actuation and sensing. A similar process in the Model-Based Design of Systems-on-a-Chip (SoCs) is outlined in [40].

Each of these transformations and calibrations introduces discrepancies between the behavior of the original system, which we generically refer to as **the nominal system** $\mathcal{M}$, and the behavior of the derived system that is produced, which we generically refer to as **the derived system** $\mathcal{I}$. These discrepancies are spatial (e.g. slightly different signal values in response to same stimulus, dropped samples, out-of-order samples) and temporal (e.g. different timing characteristics of the outputs, delayed responses due to unmodeled physical phenomena like transport delay), and their magnitude can vary as time progresses. The same situation arises when $\mathcal{I}$ is derived from $\mathcal{M}$ by a process of simplification: e.g. in Model Order Reduction (MOR), modeling from first physics principles yields a high-dimensional dynamical system $\mathcal{M}$, which takes a long time to simulate. This is then reduced to a lower-dimensional ('lower order') system $\mathcal{I}$, which is used to perform fast simulations where appropriate. Along the V process in Fig. 1, a simplifying derivation process can be seen as traversing the left branch of the V in the reverse direction from bottom to top. This raises two questions:

- First, what is the relationship between the **behaviors** of the "nominal" model $\mathcal{M}$ and "derived" model $\mathcal{I}$ (e.g. cycles 2 and 3 in Fig. 1)? Can it be quantified?
- Second, if the simpler of the two systems $\mathcal{M}$ and $\mathcal{I}$ has been formally verified to satisfy some specification, can anything be said automatically about the specifications satisfied by the more complicated one?

If the simpler model, say $\mathcal{M}$, was nondeterministic and the structure of $\mathcal{I}$ was fully known, then the answer to both questions could be established through behavioral inclusions, i.e., is it true that every behavior of $\mathcal{I}$ can be

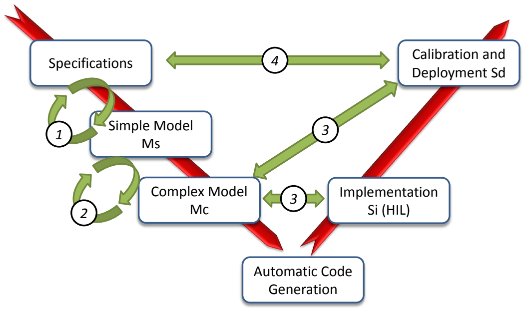

Fig. 1: Typical V process in MBD. (1) Verifying that the simple model satisfies the functional requirements; (2) Establishing a relationship between the simple and complex model; (3) Verifying conformance of implementation to the model; (4) Checking that the end product satisfies the functional requirements. Most of these steps are iterative.

exhibited by $\mathcal{M}$, in response to the same stimulus? However, in practice, non-deterministic models are rarely utilized and supported by industry tools for MBD such as LabView^{TM}, Simulink/Stateflow^{TM}, or Spice^{TM}. Moroever, irrespective of whether the derivation process has formal guarantees (such as automatic code generation in [5]), rarely do the models capture accurately all physical phenomena, so that inclusion is unlikely to hold in a realistic scenario. Similar difficulties with formal methods arising from having multiple models were outlined by [8]. Thus, in lieu of behavioral inclusion, an appropriate quantifiable notion of *closeness* between the systems behaviors is required, and this is introduced in Section II-B. If system $\mathcal{I}$ lent itself to formal methods, then the second question could also be answered by formal verification. For example in [30], a method for checking formal equivalence of a Simulink model to its generated C code is presented. However, it is not always possible to verify formally that $\mathcal{I}$ satisfies the formal specification: for example, a component purchased from a third party might allow only limited observability and not lend itself to formal methods. Or, system $\mathcal{I}$ might be too large to handle by today's formal tools. By evaluating closeness between the systems' *behaviors*, on the other hand, it is possible to draw conclusions about one from studying the behavior of the other: this is presented in Section III.

In previous work [2], [3], closeness between two output signals of two systems was mathematically formalized via the notion of $(T, J, (\tau, \varepsilon))$-closeness (Def. 2.2). This closeness measure quantitatively captures distances between two signals in both space and time, while allowing for samples from either signal to be dropped, and for signal values to be locally re-ordered. The *conformance degree* between two systems $\mathcal{M}$ and $\mathcal{I}$ was then defined via the closeness between their output signals, and *conformance testing* is then the process of calculating the conformance degree between the two systems, which was done by Simulated Annealing. In this paper, using the formalism of hybrid dynamical systems, we extend that work in four directions:

1) We refine the definition of conformance degree in Section II-B to reflect the two broad categories of derivation processes: simplification and implementation.
2) We give an automatic procedure in Section III for deriving a formal specification (over hybrid time) satisfied by the derived $\mathcal{I}$, given the formal specification satisfied by the nominal $\mathcal{M}$.
3) We argue that conformance testing can significantly alleviate the verification burden by allowing us to re-use previous testing results when the specification changes.
4) We explore the use of alternative algorithms for approximating the conformance degree between two systems in Section IV. Specifically, we explore the use of Rapidly-exploring Random Trees for arbitrary controllable systems, and the use of state-of-the-art commercial solvers for the restricted class of switched linear systems.

Experiments (Section V) and a review of related work in the literature (Section VI) conclude the paper. All proofs are in the online technical report [4].

Notation. Given an n-tuple $\alpha = (a_1, a_2, \dots a_n)$, we denote by $\mathbf{pr}_i(\alpha)$ the i-th element of the tuple, i.e., $\mathbf{pr}_i(\alpha) = a_i$. Similarly, we let $\mathbf{pr}_{i,j}(\alpha) = (a_i, a_j)$. Given a relation $R \subset A \times B$, and $b \in B$, we also define $\mathbf{pr}_b(R) = \{a \in A : (a, b) \in R\}$. The set of integers is $\mathbb{Z}$, of non-negative integers is $\mathbb{N}$, $\mathbb{N}_{>0} = \mathbb{N} \setminus \{0\}$, and the set of non-negative reals is $\mathbb{R}_+$. For $N \in \mathbb{N}$, $[N]$ is the set $\{0, \dots, N\}$. For reals a and b, we write $a \vee b = \max\{a, b\}$ and $a \wedge b = \min\{a, b\}$. For $x \in \mathbb{R}^n$, $\|x\|$ is the Euclidian norm (though any norm will do). Finally, $\#S$ is the cardinality of set S.

II. CONFORMANCE OF SYSTEMS

A. System Model

In this paper, we deal with embedded control systems. Such systems typically have certain 'modes' of operation, and the dynamics are generally different between the modes. For example, a switched power converter is a common electronic component with one switch. Depending on the switch's position, the circuit can be in one of two modes, with different dynamics depending on the active circuit elements [32]. Jumps between modes are modeled to be instantaneous. To model such systems, we adopt the hybrid systems formalism. Hybrid systems include as special cases Extended FSMs [16], switched, impulsive and classical nonlinear and linear dynamical systems, and have been used extensively to model embedded control systems. Specifically, let $H \subset \mathbb{R}^n$ be the state space, C and D be subsets of H, U be a set of input values, and F, G, and h be functions defined over H. The *hybrid dynamical system* $\mathcal{H}$ with data C, F, D, G, h, internal state $\eta \in H$ and output $y \in Y$ is governed by ($\dot{\eta}$ is the time derivative of η) [21]

$$\mathcal{H} \begin{cases} \dot{\eta} & = F(\eta, u) \qquad (\eta, u) \in C \\ \eta^+ & = G(\eta, u) \qquad (\eta, u) \in D \\ y & = h(\eta) \qquad\quad \eta \in H \end{cases} \tag{1}$$

The discrete mode can be part of the state variable η, e.g. $\eta = [x, \ell]$, where ℓ takes values in a finite set L, and x is the real-valued state of the system (e.g. voltage). In this case, $\dot{\ell} = 0$. The 'jump' map G models the change in system state at a mode change, or 'jump', and the jump set D captures the conditions causing a jump. The 'flow' map F models state evolution away from jumps, while (η, u) is in the flow set C. System trajectories start from a specified set of initial conditions $H_0 \subset H$. Finally, the output of the system $y \in Y$ is given as a function h of its internal state, and its input is

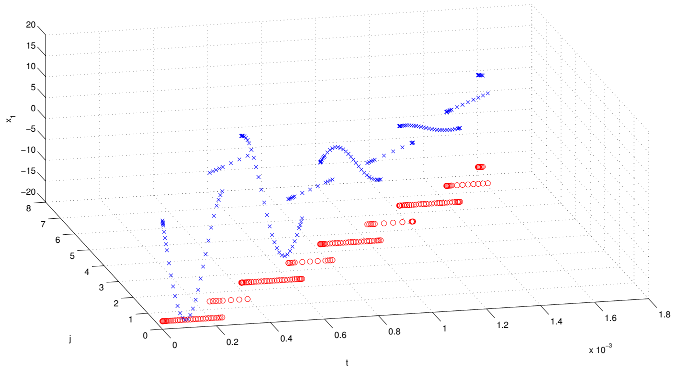

Fig. 2: Hybrid-TS for a 2-mode DC-to-DC buck-boost converter [32]. The red circles show hybrid time evolution $\mathbf{pr}_{2,3}(\mu)$ (each circle corresponds to a value $(t(i), j(i))$, while the crosses show the value of $\mathbf{pr}_1(\mu)$ (each cross corresponds to $y(i)$). Perspective distortion causes the circles to be misaligned along the j-axis. With every mode switch ('jump'), the j parameter increments by 1. Between jumps, the system evolves along the t axis as time progresses.

given by u which takes values in a set U. This is common hybrid systems terminology.

The trajectories (or 'solutions' or 'traces') of purely continuous-time dynamical systems (with only one mode) are parameterized by the time variable t, and those of purely discrete-time dynamical systems (with no continuous evolutions) are parametrized by the number of discrete jumps j. Following Goebel and Teel [22], the trajectories of hybrid systems are parameterized by both t and j, to reflect that both evolution mechanisms are present. (For example, this describes the view of time for SoC verification in [18]). The resulting time structure is referred to as 'hybrid time'. Hybrid time is better suited to capture phenomena unique to the modeling of hybrid systems, like Zeno executions [25], and to study issues related to composition of hybrid systems [37]. See also [21, Ch. 2] and references therein. We further consider that the outputs of a dynamical system are first sampled (or, in simulation, a numerical integrator generates a solution) before being fed to a controller. Thus, we model the outputs of a hybrid system as *hybrid-timed sequences*. Specifically, let $N \in \mathbb{N}_{>0}$ be a positive integer and $T \in \mathbb{R}_+$ be a positive real.

Definition 2.1: Given a set Y, a Y**-valued hybrid-timed sequence** (hybrid-TS or simply TS) is a function $\mu : \{0, 1, \ldots, N\} \to Y \times [0, T] \times \mathbb{N}$, such that for all $i \in \{0, 1, \ldots, N\}$, $\mathbf{pr}_{2,3}(\mu(i)) = (t(i), j(i))$ with $t(0) = j(0) = 0$, $t(i) \le t(i+1)$ and $j(i) \le j(i+1)$, $t(i) = t(i+1) \implies j(i) < j(i+1)$ and $j(i) = j(i+1) \implies t(i) < t(i+1)$. The **domain** of μ is $\mathbf{dom}(\mu) = \{0, 1, \ldots, N\} = [N]$.

For a TS μ, the first component, i.e., $\mathbf{pr}_1(\mu) = y$ captures the output of the system, while the second and third components, i.e., $\mathbf{pr}_{2,3}(\mu) = (t, j)$, capture the absolute time t and the number of jumps j that led to the state y. See Fig. 2. Together, (t, j) are referred to as 'hybrid time'. Most of the time, the set Y will be clear from the context.

Given an initial state $\eta \in H_0$ and an input TS $\mathfrak{u}$ (which is a U-valued TS), the system $\mathcal{H}$ produces an output Y-valued

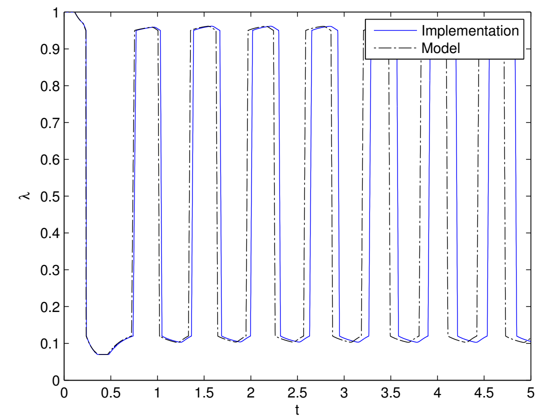

Fig. 3: The 1^{st} component $\mathbf{pr}_1(\mu)$ of an output TS of a fuel control system model $\mathcal{M}$ and its implementation $\mathcal{I}$. For each $i \in \mathbf{dom}(\mu)$, $\mathbf{pr}_1(\mu(i))$ is a real varlue λ.

TS μ, which we note as $\mu = \mathcal{H}(\eta, \mathfrak{u})$. The TS μ can be the result of a sampling process or a numerical integration. Then the sequence of 'timestamps' $\mathbf{pr}_{2,3}(\mu)$ represents the sequence of (hybrid) sampling times, or times at which a numerical solution is computed: $\mathbf{pr}_1(\mu(i))$ is the value of the output at (hybrid) time $\mathbf{pr}_{2,3}(\mu(i))$. We do not assume, in general, that the sampling period (or integration step) is constant. Note that two output TS of the same system may have different domains. We refer the reader to [22] for exact definitions of discrete and hybrid time domains, arcs and trajectories.

Assumption 2.1: We assume that when system $\mathcal{I}$ is derived (by some application-dependent process) from $\mathcal{M}$, there exists a surjective and left-total relation $R \subset H_{0,M} \times H_{0,I}$ relating the initial states of the two systems. This is commonly true in practice to enable testing; we will say **'$\mathcal{I}$ is derived from $\mathcal{M}$ with relation R'.** The output space Y is assumed equipped with a metric $\mathbf{d}$. Finally, for every initial condition $\eta_0 \in H_0$ and input TS $\mathfrak{u}$, the system $\mathcal{H}$ produces at least one output TS. This is imposed to avoid modeling issues where either system's equations have no solutions.

B. *Conformance via $(T, J, (\tau, \varepsilon))$-closeness*

In this section, we introduce the $(T, J, (\tau, \varepsilon))$-closeness measure between the output TS of hybrid systems in time and space. It is derived from [22].

Definition 2.2 ($(T, J, (\tau, \varepsilon))$-closeness): Take a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and parameters $\tau, \varepsilon > 0$. Two timed sequences $\mu = (y, t, j)$ and $\mu' = (y', t', j')$ with domains $[N]$ and $[N']$, respectively, are $(T, J, (\tau, \varepsilon))$**-close**, which we write $\mu \approx_{(\tau,\varepsilon)} \mu'$, if
(a) for all $i \in [N]$ such that $t(i) \le T, j(i) \le J$, there exists $k \in [N']$ such that $j(i) = j'(k), |t(i) - t'(k)| < \tau$, and $\|y(i) - y'(k)\| < \varepsilon$
(b) for all $i \in [N']$ such that $t'(i) \le T, j'(i) \le J$, there exists $k \in [N]$ such that $j'(i) = j(k), |t'(i) - t(k)| < \tau$, and $\|y'(i) - y(k)\| < \varepsilon$
The infimum of all ε such that μ and μ' are $(T, J, (\tau, \varepsilon))$-close is called the **achievable closeness degree given** τ.

$(T, J, (\tau, \varepsilon))$-closeness may be thought of as giving a proximity measure between the two hybrid-timed sequences, both in time and space. Allowing some 'wiggle room' in both time and space is important for conformance testing: e.g. in Fig. 3, intuitively, the two output signals are very similar, yet the sup norm would give a large value to the distance between them. Thus $(T, J, (\tau, \varepsilon))$-closeness captures nicely the intuitive notion that 'the outputs should still look alike'. The two values T and J limit our testing horizon, and will typically be set based on application domain considerations. When they are clear from the context, we will drop them to simplify the language.

Finally, Def. 2.2 requires equality in the number of jumps j between the two TS, but the results of this paper can be extended in a straightforward manner to allow some wiggle in the numbers of jumps, i.e. $|j(i) - j'(k)| < \delta$.

Definition 2.3: Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hybrid systems, such that $\mathcal{H}_2$ is derived from $\mathcal{H}_1$ with relation $R \subset H_{0,1} \times H_{0,2}$. Take a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and parameters $\tau, \varepsilon > 0$. We say that **system $\mathcal{H}_2$ simulates $\mathcal{H}_1$ with precision** (τ, ε), which is written $\mathcal{H}_1 \preceq_{\tau,\varepsilon} \mathcal{H}_2$, if for all $(\eta_1, \mathrm{u}) \in H_{0,1} \times \mathfrak{U}$, and for all $\mu_1 = \mathcal{H}_1(\eta_1, \mathrm{u})$, there exists $\eta_2 \in H_{0,2}$ s.t. $(\eta_1, \eta_2) \in R$ and for some $\mu_2 = \mathcal{H}_2(\eta_2, \mathrm{u})$, $\mu_1 \approx_{(\tau,\varepsilon)} \mu_2$.

This definition is near-identical to that of approximate simulation given in [26, Def. 2.6]. The subtle but important differences due to our setting are that : 1) the relation R between initial sets does not arise here as a result of the approximation by (τ, ε)-closeness, rather it is dictated by the derivation process from $\mathcal{M}$ to $\mathcal{I}$. This bounds the quality of the approximation. 2) Whereas in [26], R is required to be left-total only, here we require R to be surjective as well. This again is dictated by the derivation process. Modulo this distinction, our work fits within the approximate bisimulation framework presented in [26]. Therefore, we use the same terminology ('simulation') and notation.

From a conformance perspective, it is preferable to have a smaller ε and a smaller τ. Since only a partial order exists on the (τ, ε) pairs, we define 'partial' conformance degrees between systems.

Definition 2.4: Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hybrid systems. The **conformance degree of $\mathcal{H}_1$ to $\mathcal{H}_2$ given** τ is defined as the smallest ε such that $\mathcal{H}_1 \preceq_{\tau,\varepsilon} \mathcal{H}_2$:

$$\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2) := \inf\{\varepsilon : \mathcal{H}_1 \preceq_{\tau,\varepsilon} \mathcal{H}_2\}$$

An obvious analogous definition holds for conformance degree given ε. Thereafter, we will always be referring to the conformance degree given τ and drop 'given τ' from the terminology. Note that because the conformance degree is defined using the output behaviors of the systems, and not their internal structures, observability limitations on either $\mathcal{I}$ or $\mathcal{M}$ do not affect our ability to compute it.

Example 1 (Power converters): Power converters are common electronic components, used in many safety-critical systems. A DC-to-DC converter accepts an input DC voltage V_s and converts it to a reference V_{ref}. It has two modes, and the switch between them is software-controlled [32]. We use a simplified model of a power converter as a hybrid system in Section V, and use this model to compute the (τ, ε)-closeness between a model and its implementation. □

Example 2 (Implementation process): A controller is developed for an automatic transmission model in Simulink. Controller code is then automatically generated by Simulink, targeting a given computational platform, like an embedded board. Because the board has different computation precision than the general-purpose host on which the model was verified, and because Hardware-In-the-Loop testing introduces delays and unmodeled interrupts, the generated code+automatic transmission closed-loop system ($\mathcal{I}$) will produce outputs that are different from the Simulink model+automatic transmission ($\mathcal{M}$). Conformance testing is needed to quantify the discrepancy between the two systems, and to derive what specification is satisfied by $\mathcal{I}$, given that $\mathcal{M}$ satisfies its specification. □

In all the above scenarios, we wish to test the simplified system, say, $\mathcal{I}$, rather than the costly system, say, $\mathcal{M}$. In particular, if we check that $\mathcal{I}$ satisfies some property φ (which we can do relatively cheaply), we wish to automatically derive a corresponding formula satisfied by $\mathcal{M}$, without checking it explicitly (which might not be possible). The result from the next section allows us to do so, *if* we know the conformance degree of $\mathcal{I}$ to $\mathcal{M}$.

C. Local disorder in $(T, J, (\tau, \varepsilon))$-close signals

A distinguishing feature of $(T, J, (\tau, \varepsilon))$-closeness as a measure of closeness between TS is that it allows for **local disorder** in the signal values: i.e. given two TS $\mu = (y, t, j)$ and $\mu' = (y', t', j')$ define the relation $\rho \subset [N] \times [N']$ by $(i, i') \in \rho$ iff $\|y(i) - y'(i')\| < \varepsilon$, $|t(i) - t'(i')| < \tau$ and $j(i) = j'(i')$. Then there may exist $(i, i') \in \rho$ and $(k, k') \in \rho$ with $i < k$ and $i' > k'$. Figure 4 (top) gives a generic illustration of such a case.

We should note that all four points i, i', k, k' must occur within a window of size τ, which is why we call this *local* disorder. The pattern of Fig. 4 (top) can not repeat in consecutive windows of width τ: as shown in Fig. 4 (bottom), consecutive repetitions (indicated by the brackets) actually yield two TS whose values ($\mathbf{pr}_1(\mu)$) are merely shifted with respect to each other, as indicated by the arrows relating ρ-related samples.

Local disorder could arise in any situation where the output signal $\mathbf{pr}_1(\mu)$ of the system is distorted by noise. E.g. if the model $\mathcal{M}$ of an electric circuit produces a noise-free μ, its implentation $\mathcal{I}$ will in general suffer from parasitics and other noise sources. More generally, recall that signal values (i.e. $\mathbf{pr}_1(\mu)$) are real-valued outputs of the system, and not simply discrete 'events' whose order must be preserved. A priori, and without further defining the derivation process, there is no reason to assume that a valid derivation will preserve signal values order locally, even if globally, the nominal and derived system have similar outputs. A notion of closeness between real-valued outputs, therefore, should a priori account for (and quantify) local disorder. Thus, by allowing local disorder, $(T, J, (\tau, \varepsilon))$-closeness is well-adapted to a wider class of implementations and distortions than the measures surveyed in the literature (Section VI).

The above discussion has a consequence for the design process where $\mathcal{M}$ is implemented as $\mathcal{I}$, and $\mathcal{I}$ is deployed: if we calculate the conformance degree $\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2)$ given some $\tau > 0$, we are effectively saying that local disorder within a τ window is permissible, and should be quantified, rather

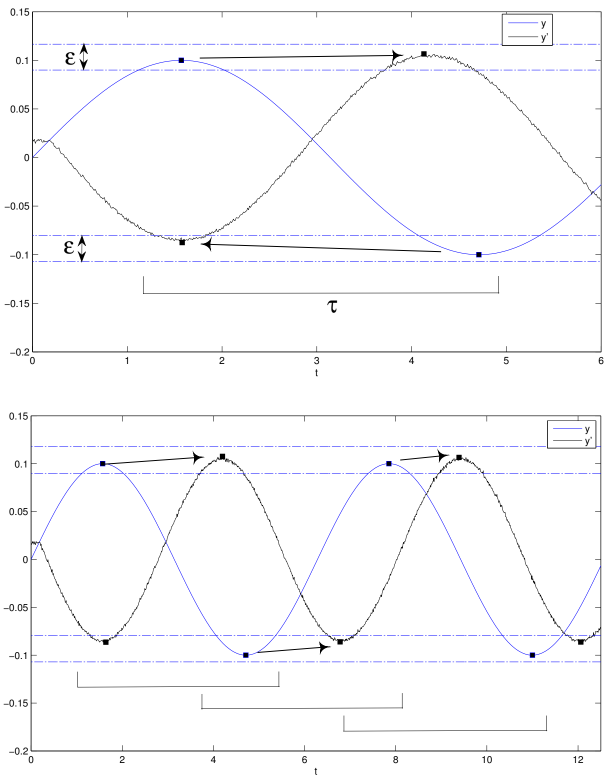

Fig. 4: Top: local disorder. The squares indicate elements of the (τ, ε)-close TS, the continuous plots are only there to show the subtending sampled signals. Samples related by ρ are related graphically by arrows. Bottom: local disorder only lasts for an interval of τ.

than flagged as an error. This makes design sense only if the temporal logic specification according to which $\mathcal{M}$ is designed contains timing intervals of width at least τ. In the next section, we further quantify the relation between satisfied properties and conformance degree.

III. Transfer of properties

In Model-Based Design (MBD), the model $\mathcal{M}$ is designed in an iterative fashion to satisfy a certain specification φ. In this work our focus is exclusively on formal specifications expressed in Metric Temporal Logic (MTL) (see Section III-A). When moving from Model testing to Implementation testing, the main question is: despite the inaccuracies introduced by the implementation process, does my Implementation $\mathcal{I}$ still satisfy the specification φ?

As mentioned in the Introduction, often, it might not be possible to formally verify φ on the more complex of the two systems, say $\mathcal{M}$. For all these reasons, our confidence in the more complex system must derive from two things: the fact that $\mathcal{I}$ satisfies φ; and that the two systems $\mathcal{M}$ and $\mathcal{I}$ have 'close' behaviors. In this section, we formalize the relation between closeness of behaviors and formula satisfiability by deriving, automatically, which formulae are satsified by a TS μ' which is (τ, ε)-close to a TS μ, given that the latter satisfies φ. Note that this *does not require any testing of* μ': the formulae are derived automatically via syntactic manipulations.

A. MTL for Hybrid Timed State Sequences

In order to introduce the MTL-based design framework in Section III, we now briefly go over the definition of Metric Temporal Logic (MTL) [29]. MTL is a temporal logic for expressing real-time properties of embedded and cyber-physical systems, and allows the specification of constraints on the timing of events. In this section, we present an extension of MTL over hybrid time. In a hybrid time domain, the time variable takes values in $\mathbb{T} = [0, T] \times \{0, \ldots, J\}$. This extension naturally subsumes the case of real-valued time. A hybrid time set is a non-empty set of the form $\mathbb{I} = E_c \times E_d \subset \mathbb{R} \times \mathbb{N}$, where E_c is an interval in $\mathbb{R}$ and E_d is a set of successive integers. Given the hybrid time $(s, j) \in \mathbb{R} \times \mathbb{N}$, $(s, j) \oplus \mathbb{I} := \{(s', j') \mid \exists(\bar{s}, \bar{j}) \in \mathbb{I} \;.\; s' = s + \bar{s} \text{ and } j' = j + \bar{j}\}$. This is itself a hybrid time set.

Definition 3.1 (MTL$^+$ Syntax): Let AP be a set of atomic propositions and $\mathbb{I}$ be a hybrid time set. The set MTL^+ of all well-formed MTL formulas in *negation normal form* is inductively defined as $\varphi \;:=\; \top \mid \bot \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathcal{U}_{\mathbb{I}} \varphi \mid \varphi \mathcal{R}_{\mathbb{I}} \varphi$, where $p \in AP$, $\top$ is *true* and $\bot$ is *false*.

We instantiate the definitions of the semantics over abstractions of the output TS of the hybrid system $\mathcal{H}$ with respect to the sets $\mathcal{O}(p) \subseteq Y$ for all $p \in AP$. Let $(\mu, i) \models_{\mathcal{O}} \varphi$ denote the satisfaction of the MTL formula φ over a TS μ starting at sample i with respect to the atomic proposition-mapping $\mathcal{O}$. If μ does not satisfy φ under the map $\mathcal{O}$, then we write $(\mu, i) \not\models_{\mathcal{O}} \varphi$.

Definition 3.2 (MTL$^+$ Semantics): Let μ be a TS and $\mathcal{O} : AP \to \mathcal{P}(Y)$. For $i, k, l \in \mathbb{N}$, the semantics of any MTL^+ formula φ can be recursively defined as:

$$
\begin{aligned}
&(\mu, i) \models_{\mathcal{O}} \top \text{ and } (\mu, i) \not\models_{\mathcal{O}} \bot \\
&(\mu, i) \models_{\mathcal{O}} p \text{ iff } \mathbf{pr}_1(\mu(i)) \in \mathcal{O}(p) \\
&(\mu, i) \models_{\mathcal{O}} \neg p \text{ iff } \mathbf{pr}_1(\mu(i)) \notin \mathcal{O}(p) \\
&(\mu, i) \models_{\mathcal{O}} \varphi_1 \vee \varphi_2 \text{ iff } (\mu, i) \models_{\mathcal{O}} \varphi_1 \text{ or } (\mu, i) \models_{\mathcal{O}} \varphi_2 \\
&(\mu, i) \models_{\mathcal{O}} \varphi_1 \wedge \varphi_2 \text{ iff } (\mu, i) \models_{\mathcal{O}} \varphi_1 \text{ and } (\mu, i) \models_{\mathcal{O}} \varphi_2 \\
&(\mu, i) \models_{\mathcal{O}} \varphi_1 \mathcal{U}_{\mathbb{I}} \varphi_2 \text{ iff } \exists k \in \mathbf{dom}(\mu) \text{ such that} \\
&\quad \mathbf{pr}_{2,3}(\mu(k)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I} \text{ and } (\mu, k) \models_{\mathcal{O}} \varphi_2 \\
&\quad \text{and } \forall l \text{ with } i \le l < k \text{ we have } (\mu, l) \models_{\mathcal{O}} \varphi_1 \\
&(\mu, i) \models_{\mathcal{O}} \varphi_1 \mathcal{R}_{\mathbb{I}} \varphi_2 \text{ iff } \forall k \ge i, \\
&\quad \mathbf{pr}_{2,3}(\mu(k)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I} \text{ implies } (\mu, k) \models_{\mathcal{O}} \varphi_2 \\
&\quad \text{or } \exists l \text{ with } i \le l < k \text{ such that } (\mu, l) \models_{\mathcal{O}} \varphi_1
\end{aligned}
$$

Other operators can be defined using the above, e.g. the Eventually operator $\Diamond_{\mathbb{I}} \varphi := \top \mathcal{U}_{\mathbb{I}} \varphi$ and the Always operator $\Box_{\mathbb{I}} \varphi := \bot \mathcal{R}_{\mathbb{I}} \varphi$. The usual MTL$^+$ logic over real time is recovered by choosing all hybrid time sets to be $\mathbb{I} = E_c \times \mathbb{N}$.

Note that in defining the semantics for Until, we did not require that $k \ge i$, since we are allowing negative endpoints on the intervals E_c of a hybrid time set. In the case of a negative endpoint, there may be a need to refer to a state $\mu(k)$ that *preceded* $\mu(i)$.

B. Property transfer

Given a set $S \subset \mathbb{R}^n$ equipped with a metric d, $\mathcal{P}(S)$ is the set of subsets of S. Its δ-expansion $E(S, \delta)$ and δ-contraction

$C(S,\delta)$ are defined by: $E(S,\delta)=\{x\in\mathbb{R}^n \mid \inf_{s\in S} d(x,s)\le \delta\}$ and $C(S,\delta)=\mathbb{R}^n\setminus E(\mathbb{R}^n\setminus S,\delta)$. Finally, for a hybrid time set $\mathbb{I}=E_c\times E_d$ and reals a,b, define $\mathbb{I}_{\langle a,b\rangle} := (\inf E_c + a, \sup E_c + b)\times E_d$, where inf and sup are the greatest lower bound, and least upper bound, operators, respectively.

Theorem 1: Let φ be an MTL^+ formula with atomic propositions in AP and $\mathcal{O}: AP\to\mathcal{P}(Y)$. Let $\mu=(y,t,j)$, $\mu'=(y',t',j')$ be two TS such that $\mu\approx_{(\tau,\varepsilon)}\mu'$. If $(\mu,i)\models_{\mathcal{O}}\varphi$ then for all $i'\in\mathbf{dom}(\mu')$ s.t. $|t'(i')-t(i)|\le\tau$, $j(i)=j'(i')$, and $\|y(i)-y'(i')\|\le\varepsilon$,

$$(\mu',i')\models_{\mathcal{O}_\varepsilon}[\varphi]_\tau$$

where the operator $[\cdot]_\tau : MTL^+ \to MTL^+$ obeys the following rules:

$$\begin{aligned}
[\top]_\tau=\top \quad &, \quad [\bot]_\tau=\bot\\
[p]_\tau=p^+ \quad &, \quad [\neg p]_\tau=p^-\\
[\varphi_1\vee\varphi_2]_\tau &= [\varphi_1]_\tau\vee[\varphi_2]_\tau\\
[\varphi_1\wedge\varphi_2]_\tau &= [\varphi_1]_\tau\wedge[\varphi_2]_\tau\\
[\varphi_1\mathcal{U}_{\mathbb{I}}\varphi_2]_\tau &= (\Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau)\\
&\quad \mathcal{U}_{\mathbb{I}_{\langle-2\tau,2\tau\rangle}}(\Diamond_{[0,2\tau)\times\{0\}}[\varphi_2]_\tau)\\
[\varphi_1\mathcal{R}_{\mathbb{I}}\varphi_2]_\tau &= (\Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau)\\
&\quad \mathcal{R}_{\mathbb{I}_{\langle2\tau,-2\tau\rangle}}(\Diamond_{[0,2\tau)\times\{0\}}[\varphi_2]_\tau)
\end{aligned}$$

where $\mathbb{I}=E_c\times E_d$ is a hybrid time set. Also, $\mathcal{O}_\varepsilon(p^+)=E(\mathcal{O}(p),\varepsilon)$ and $\mathcal{O}_\varepsilon(p^-)=C(\mathcal{O}(p),\varepsilon)$.

The proof is in the technical report [4]. The results of [24] and [35] can now be recovered as special cases of the above theorem. The result of [24] is a special case of Thm. 1 where only time is allowed to deviate ($\varepsilon=0$). The result of [35] requires an order-preserving notion of closeness (which it calls "order-preserving ε-retiming"). Both operate over real time, rather than hybrid time, which is more suitable for the study of hybrid systems. To illustrate the content of Thm. 1, we give two examples:

$$\begin{aligned}
[\Box_{[3,6]\times\{1,2\}}p]_\tau &= [\bot\mathcal{R}_{\mathbb{I}}p]_\tau\\
&= \Diamond_{(-2\tau,0]}\bot\mathcal{R}_{[3+2\tau,6-2\tau]\times\{1,2\}}\Diamond_{[0,2\tau)}p^+\\
&= \bot\mathcal{R}_{[3+2\tau,6-2\tau]\times\{1,2\}}\Diamond_{[0,2\tau)}p^+\\
&= \Box_{\mathbb{I}_{\langle-2\tau,2\tau\rangle}}(\Diamond_{[0,2\tau)}[p]_\tau)\\
[\Diamond_{\mathbb{I}}\varphi]_\tau &= \Diamond_{\mathbb{I}_{\langle-2\tau,4\tau\rangle}}[\varphi]_\tau
\end{aligned}$$

The main result of this section now follows from the definitions and Thm. 1, and its proof is in [4]. For a hybrid system $\mathcal{H}$ and a map $\mathcal{O}: AP\to\mathcal{P}(H)$, we write $\mathcal{H}^\tau\models_{\mathcal{O}}\varphi$, if for all output TS μ of $\mathcal{H}$, there exists $i\in\mathbf{dom}(\mu)$ s.t. $\mathbf{pr}_2(\mu(i))\le\tau$ and $(\mu,i)\models_{\mathcal{O}}\varphi$. We simply write $\mathcal{H}\models_{\mathcal{O}}\varphi$ if $\mathcal{H}^0\models_{\mathcal{O}}\varphi$.

Theorem 2: Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hybrid systems, and φ be an MTL^+ formula. If $\mathcal{H}_1\preceq_{\tau,\varepsilon}\mathcal{H}_2$ and $\mathcal{H}_2\models_{\mathcal{O}}\varphi$, then $\mathcal{H}_1^\tau\models_{\mathcal{O}_\varepsilon}[\varphi]_\tau$.

The theorem may be interpreted informally as saying that system $\mathcal{H}_1$ needs an 'initialization phase', of duration at most τ, before it satisfies $[\varphi]_\tau$. The role played by the Eventually operators with negative time intervals appearing in $[\varphi_1\mathcal{U}_{\mathbb{I}}\varphi_2]_\tau$ and $[\varphi_1\mathcal{R}_{\mathbb{I}}\varphi_2]_\tau$ of Thm. 1 also becomes clear: they serve to cover this initialization phase.

If, say, $\mathcal{M}$ is what ultimately gets deployed (or is input to the next phase of the design cycle), and $\mathcal{I}$ is derived from $\mathcal{M}$ by a simplification for testing purposes (e.g. model order reduction), then we care about $\mathcal{M}$ verifying the specification φ_s, but we want to do the testing on $\mathcal{I}$ since it is simpler. We then use Thm. 2 to derive the specification $[\varphi_p]_\tau$ satsisfied by $\mathcal{M}$, and whether it equals φ_s. Thus in this case, we identify $\mathcal{H}_1=\mathcal{M}$ and $\mathcal{H}_2=\mathcal{I}$ in Thm. 2. If, as often happens, a new specification becomes relevant, then instead of testing the expensive $\mathcal{M}$, we may simply test $\mathcal{I}$, and use Thm. 2 to conclude the specification satisfied by $\mathcal{M}$. Thus conformance testing is a one-time cost (as long as $\mathcal{M}$ isn't modified), which reduces the testing effort when specifications change.

IV. Computing the conformance degree

In this section we treat the problem of computing the conformance degree given in Def. 2.4. Conformance testing is the process of finding two trajectories μ_1 and μ_2, of $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, such that they achieve $(\tau,\mathbf{CD}_\tau(\mathcal{H}_1,\mathcal{H}_2))$-closeness. The result can be used in two ways: first, the conformance degree is needed to apply the property transfer results of the previous section. Secondly, μ_1 and μ_2 can be used to debug a derivation process: suppose $\mathcal{M}$ and $\mathcal{I}$ were designed to achieve a certain (τ,ε). If conformance testing yields an achievable degree (τ,ε') with $\varepsilon'>\varepsilon$, i.e. the true distance is greater than what was designed for, then the 'witness' TS μ_1 and μ_2 act as debugging traces to detect where the behavior was erroneous, and therefore what needs to be fixed in the derivation process. The details of such debugging are naturally application-dependent.

The value $\mathbf{CD}_\tau(\mathcal{H}_1,\mathcal{H}_2)$ is computed in stages. For two TS $\mu=(y,t,j)$ and $\mu'=(y',t',j')$, define

$$cd(\mu,\mu') := \max\{cd_1(\mu,\mu'), cd_1(\mu',\mu)\}$$

where

$$cd_1(\mu,\mu') = \max_{i\in\mathbf{dom}(\mu)} \min_{\substack{i'\in\mathbf{dom}(\mu'):\\ j(i)=j'(i')\\ |t(i)-t'(i')|<\tau}} \|y(i)-y'(i')\|$$

Note that cd is symmetric. Then, given two hybrid systems related by R, $\mathbf{CD}_\tau(\mathcal{H}_1,\mathcal{H}_2)$ is calculated as

$$\begin{aligned}
\mathbf{CD}_\tau(\mathcal{H}_1,\mathcal{H}_2) &= \sup\{cd(\mathcal{H}_1(\eta_1,\mathfrak{u}),\mathcal{H}_2(\eta_2,\mathfrak{u})):\\
&\qquad \eta_1\in H_{0,1},\mathfrak{u}\in\mathfrak{U},\eta_2\in H_{0,2}\cap\mathbf{pr}_{\eta_1}(R)\}
\end{aligned}$$

This dynamically-constrained optimization can be seen to be nonsmooth, nonlinear and indeed in general nonconvex. Its format does not satisfy the principle of optimality because of the max operators and so it does not lend itself to dynamic programming. It doesn't take the form of an integrated or final cost, and so is not readily amenable to optimal control methods. In our previous work [3], due to these complexities, we adopted Simulated Annealing (SA) as a general-purpose, derivative-free, stochastic global optimizer. We will next develop the computation of $\mathbf{CD}_\tau(\mathcal{H}_1,\mathcal{H}_2)$ in two directions: by the use of an adapted Rapidly-exploring Random Trees (RRTs) [13], and by computing an upper bound in the case of switched linear systems, which we derive now.

Let z be a symbol denoting a pair of TS: $z = (\mu_1, \mu_2) \in \mathcal{Z}$,

$$\begin{aligned}\mathcal{Z} \;=\; & \{(\mu_1, \mu_2) : \mu_2 = \mathcal{H}_2(\eta_2, \mathfrak{u}), \mu_1 = \mathcal{H}_1(\eta_1, \mathfrak{u}) \\ & \text{s.t. } (\eta_1, \mathfrak{u}) \in H_{0,1} \times \mathfrak{U}, \eta_2 \in H_{0,2} \cap \mathbf{pr}_{\eta_1}(R)\}\end{aligned}$$

Noting that $\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2) = \sup_{z\in\mathcal{Z}} cd_1(\mu_1, \mu_2) \vee \sup_{z\in\mathcal{Z}} cd_1(\mu_2, \mu_1)$, we compute $\sup_{z\in\mathcal{Z}} cd_1(\mu_1, \mu_2) := \varepsilon_1^*$ and $\sup_{z\in\mathcal{Z}} cd_1(\mu_2, \mu_1)$ separately. These two optimizations can be done in parallel and are symmetric in their structure, so in the remainder we focus on ε_1^*.

Proposition 4.1: For each $z = (\mu_1, \mu_2) \in \mathcal{Z}$, with $\mu_1 = (y_1, t_1, j_1), \mu_2 = (y_2, t_2, j_2)$, $i \in \mathbf{dom}(\mu_1)$, define the set $S(i, z) := \{k \in \mathbb{Z} \mid i + k \in \mathbf{dom}(\mu_2)\}$. If $S := \cap_{z=(\mu_1,\mu_2)\in\mathcal{Z}, i\in\mathbf{dom}(\mu_1)} S(i, z) \neq \emptyset$, define for each $k \in S$, $g_k(z) = \max_{i\in\mathbf{dom}(\mu_1)} \|y_1(i) - y_2(i + k)\|^2$. Then $\varepsilon_1^* \leq K := \sqrt{\min_{k\in S} \sup_{z\in\mathcal{Z}} g_k(z)}$

The proof is in [4]. The set S contains indices for which g_k is a well-defined function of z, and thus needs to be non-empty. While in general, the non-emptiness hypothesis might be unrealistic, it holds in the important case of switched linear systems treated in Section IV-B.

A. Rapidly-exploring Random Trees

RRT is a very popular and efficient method of robot motion planning (see [13] and references therein). Its strength lies in its ability to explore the robot space quickly to reach a target from a given starting point. In this section we present an adaptation of RRTs to the problem of computing $\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2)$. The *workspace* $\mathcal{Q}$ of the RRT is the product of the two output spaces Y_1 and Y_2 of nominal system $\mathcal{H}_1$ and derived $\mathcal{H}_2$: $\mathcal{Q} = Y_1 \times Y_2$. Let $\mathbf{dist}_\mathcal{Q} : \mathcal{Q} \times \mathcal{Q} \to \mathbb{R}_+$ be a distance function over $\mathcal{Q}$.

RRT builds a tree to explore the workspace. The root of the tree is chosen to be a pre-determined couple of initial outputs, namely $q_0 = [h_1(\eta_1), h_2(\eta_2)]$, with $(\eta_1, \eta_2) \in R$. Suppose the tree currently has $i \geq 1$ nodes. A probability distribution with support $\mathcal{Q}$ is used to select a sample $q_s = [y_1, y_2]$ in the workspace. The nearest $\mathbf{dist}_\mathcal{Q}$-neighbor to q_s on the tree is found, say $q_{near} = [y_1^{near}, y_2^{near}]$. See Fig. 5. A local controller is then applied to $\mathcal{H}_2$ to synthesize an input TS $\mathfrak{u}_i$, of duration D_{plan}, which drives $\mathcal{H}_2$ from y_2^{near} to y_2. This input is applied for a pre-determined duration D_{hor}, called the control horizon, which may be different from D_{plan}. This leads $\mathcal{H}_2$ to output y_2' (which is not necessarily equal to y_2). The same input is then applied to $\mathcal{H}_1$ which then reaches output y_1'. The new configuration $q_{i+1} = [y_1', y_2']$ is then added to the tree, and the process repeats until the tree has a pre-determined size, or some measure of coverage exceeds a specified threshold [14]. Once the tree is constructed, every branch from root to leaf represents an evolution of the two systems, starting from (η_1, η_2), and under a series of common input TS. So we can associate a pair of TS $\mu = \mathcal{H}_1(\eta_1, \mathfrak{u})$ and $\mu' = \mathcal{H}_2(\eta_2, \mathfrak{u})$ to each branch, and compute $cd(\mu, \mu')$ along that branch. The largest computed cd-value among all the branches constitutes an estimate of $\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2)$ (more accurately, it is an under-approximation).

Guarantees of this method derive from the guarantees provided by the underlying RRT algorithm - see for example [28]. This modified RRT only assumes that the systems have controllers: no other assumption is made concerning their structure or properties.

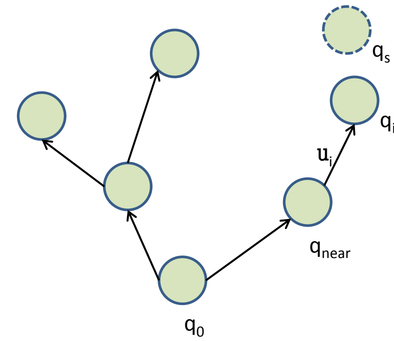

Fig. 5: RRT for computing the conformance degree.

B. Switched linear systems

In this section we show how the upper bound of Prop. 4.1 can be computed when both systems are switched linear systems driven by an external switching signal.

Assumption 4.1: Both $\mathcal{M}$ and $\mathcal{I}$ are switched linear systems (defined below); these models arise frequently in supervisory control of linear systems. The integration step or sampling period is constant: $t(i+1) - t(i) = t'(i+1) - t'(i) = \delta t > 0\ \forall i$. E.g. the output may be measured via a sample-and-hold circuit. The initial sets $H_{0,M}$ and $H_{0,I}$ and the state spaces H_M and H_I are bounded boxes in $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$, respectively. There exists a linear transformation V_R between the initial state η_I of $\mathcal{H}_I$ and that η_M of $\mathcal{H}_M$: $\eta_I = V_R \cdot \eta_M$. E.g. this is true whenever $\mathcal{I}$ is obtained by MOR from a switched linear $\mathcal{M}$.

A switched linear system $\mathcal{H}$ is a hybrid system (1) where the flow and output functions are linear with respect to the state and the input, and there are no resets of the state (i.e. G is the identity). It can be seen as a collection of L linear sub-systems $(A_\ell, B_\ell, C_\ell, D_\ell)$, $\ell \in [L], L \in \mathbb{N}_{>0}$, described by the discrete-time equations:

$$\mathcal{H} : \begin{cases} \eta(0) & \in H_0 \subset \mathbb{R}^n \\ \eta(s+1) & = A_{a(s)}\eta(s) + B_{a(s)}u(s) \\ y(s) & = C_{a(s)}\eta(s) + D_{a(s)}u(s) \end{cases} \tag{2}$$

where $a : [0, T] \to [L]$ is the piece-wise constant right-continuous external switching signal with left limits, and finitely many discontinuities in any bounded interval. When $a(s)$ changes value, the system starts obeying the dynamics of the new mode, and hybrid time advances by increasing j. An output TS of $\mathcal{H}$ is $\mu = (y, t, j)$ s.t. $t(i) = i \cdot \delta t$, $y(i) = h(\eta(t(i)))$, and $j(i) = \#\{s \in [0, i \cdot \delta t] \mid a \text{ is discontinuous at } s\}$. Both $\mathcal{M}$ and $\mathcal{I}$ are described by (2) with common switching signal, input TS, mode set $[L]$, output set Y, and (naturally) different matrices $(A_\ell, B_\ell, C_\ell, D_\ell)$. In particular, this implies that their output TS have the same domain, and that $\mathbf{pr}_{2,3}(\mu_M) = \mathbf{pr}_{2,3}(\mu_I) = (t, j)$, with t and j given above. Therefore in this sub-section, we identify η of (2) with the first component of the TS (η, t, j), and similarly $a \equiv \mathbf{pr}_1((a, t, j))$, where (t, j) is the common domain of all the TS.

We now present the elements of the formal ODE-constrained optimization problem we seek to solve. The search variable of the optimization is simply the two output TS of the two systems, 'unrolled' over $N+1$ time steps, starting from corresponding initial conditions, and subject to the same input signal $\mathfrak{u}$. Formally, the search variable z can now be written as a vector of samples:

$$z = [\eta_1(0), \eta_1(1), \ldots, \eta_1(N), \eta_2(0), \eta_2(1), \ldots, \eta_2(N)] \in \mathcal{Z}$$

Note that the initial states of the two systems are part of the search variable. If we wish to make the input TS $\mathfrak{u}$ part of the search, we may similarly unroll it and append its sampled values to the search vector z.

Putting it all together, our optimization problem is:

$$\begin{aligned}
\max_{z} \quad & g_k(z) = \max_{i \in \mathbf{dom}(\mu_1)} \|y_1(i) - y_2(i+k)\|^2 \qquad (3)\\
\text{s.t.} \quad & \text{(Space Constraint) } \forall\, i = 0, \ldots, N\\
& lb_{i,1} \le \eta_1(i) \le ub_{i,1},\ lb_{i,2} \le \eta_2(i) \le ub_{i,2}\\
& \text{(Output constraint) } \forall\, i = 0, \ldots, N\\
& y_1(i) = C_{1,a(i)}\eta_1(i) + D_{1,a(i)}u(i)\\
& y_2(i) = C_{2,a(i)}\eta_2(i) + D_{2,a(i)}u(i)\\
& \text{(Dynamical constraint) } \forall\, i = 0, \ldots, N-1\\
& \eta_1(i+1) = A_{2,a(i)}\eta_1(i) + B_{1,a(i)}u(i)\\
& \eta_2(i+1) = A_{1,a(i)}\eta_2(i) + B_{2,a(i)}u(i)\\
& \text{(Implementation constraint)}\\
& \eta_1(0) = V_R \cdot \eta_2(0)
\end{aligned}$$

Proposition 4.2: If Assumption 4.1 holds, then $\mathbf{dom}(\mu) = \mathbf{dom}(\mu') = [N]$ for all $(\mu, \mu') \in \mathcal{Z}$, and for all $k \in [N]$, the function $z \mapsto g_k(z)$ is convex.

The proof is in [4]. Thus, because we are maximizing a convex function over a convex domain, it suffices to restrict the search to the feasible set's boundary. We conclude by noting that for the solution of (3) to be acceptable as valid output TS, we set the error tolerance to be less than the integration error incurred when simulating the system by numerical integration.

V. Experiments

In this section, we illustrate the preceding theory and algorithms on benchmark examples. For the first two systems, we used the state-of-the-art optimization solver KNITRO [43]. KNITRO can handle very large-scale mixed integer nonlinear programs. While not designed for nonsmooth optimization, it can still provide a number of local maxima, so we can approximate the global maximum via multi-start. To illustrate Thm. 2, we use two tools for property verification: the first is SpaceEx, a reachability analysis tool which over-approximates the reachable set of a hybrid linear system, and thus can be used to rigorously verify safety properties [17]. The second tool is S-TaLiRo, which searches the set of initial conditions and input TS (if any) for a *falsifier*, i.e. an output TS which does *not* satisfy the property [6]. S-TaLiRo can handle arbitrary MTL specifications (not just safety/reachability). Its guarantees are probabilistic: i.e. if S-TaLiRo does not find a falsifier, then we know with high probability that one does not exist. The exact probability depends on the tool's runtime and certain other parameters. Other verification methods exist like coverage-based testing, which can cover the set of initial conditions with a finite number of tests [27]. In this section, to avoid overloading the notation, a hybrid time set of the form $\mathbb{I} = E_c \times \{0\}$ will be written simply as E_c.

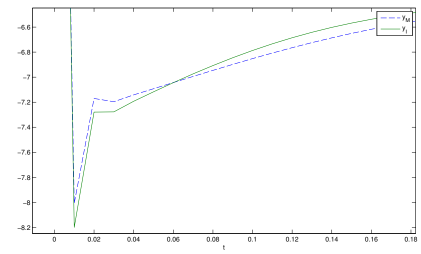

Fig. 6: Trajectories that maximize the upper bound for RLC600, zoomed in to show differences on the order of K.

RLC circuits: The first system, RLC200, is a 200D RLC circuit obtained from [23]. We take RLC200 to be the nominal model $\mathcal{M}$. We obtain $\mathcal{I}$ from $\mathcal{M}$ by balanced model order reduction (MOR), which produces a 14D linear system. Because it satisfies Assumption 4.1, we formulate the optimization as given in (3), for a given pre-determined input TS. This yields a K upper bound value (Prop. 4.2) of 0.5453. We computed the achievable closeness degree ε between the two trajectories that maximize K (i.e. the solution of (3)), and the obtained value was also 0.5453. So for this maximum, the bound K is tight. We also ran the same procedure on a 600-dimensional scaling up of RLC200 with similar results. See Fig. 6. For both systems, it took KNITRO an average of 30mins to reach a maximum.

As an example specification for RLC200, consider the following progressive settling time formula expressed in MTL: $\varphi = (\Box_{[0,0.8]}|y_1 - y_2| \le 1) \wedge (\Box_{[0.8,2.5]}|y_1 - y_2| \le 0.5)$. This formula says that in the initial 0.8 secs, the output of the reduced order system $\mathcal{I}$ must not differ from that of $\mathcal{M}$ by more than 1 Volt. Then, and up to time 2.5 secs, it must differ by even less, namely 0.5V. This reflects the gradual disappearance of transients in the circuits and settling to steady-state operation. We ran S-TaLiRo on $\mathcal{I}$, to test whether it satisfied φ. S-TaLiRo found no falsifiers, indicating that with high probability, $\mathcal{I}$ satisfies the property. The corresponding transformed formula is $(\Box_{[0.06,0.74]}\Diamond_{[0,0.06]}|y_1 - y_2| \le 1.54) \wedge (\Box_{[0.86,2.44]}\Diamond_{[0,0.06]}|y_1 - y_2| \le 1.04)$ S-TaLiRo returned no falsifiers of $[\varphi]_\tau$ by $\mathcal{M}$. □

Buck converter [32]: A DC-to-DC buck converter accepts an input DC voltage V_s and converts it down to a lower V_{ref}. It has two modes. Given a switching period P and a duty cycle f, it is in mode 1 for $f \cdot P$ units of time and mode 2 for $(1-f)P$ units. For this example's purposes, we adopt a simple open-loop strategy where the duty cycle is a function of the reference voltage: $f = V_{ref}/V_s$. When implemented, the circuit's R, L, C parameters will typically deviate from their nominal values, and the switching period P computed by the software will drift from its nominal value. Thus to study worst-case behavior, the nominal system $\mathcal{M}$ and derived $\mathcal{I}$ are taken to correspond to the two extremes of the valid ranges of R, L, C, P. This is now an example of a switched system, so we ran KNITRO to find the conformance degree given $\tau = 3e-5$ secs. It returned $\varepsilon = 2.24$ in under 4 secs. We then ran SpaceEx to verify a safety property $\varphi := \Box_{[0.001,0.0147]\times A}|y - 5| \le 1$ of $\mathcal{M}$, with $A = [\lceil (0.0147 - 0.001)/P \rceil]$. The corresponding transformed formula

$$[\varphi]_{3e-5} = \square_{[0.001+2\tau, 0.0147-2\tau]\times A} \diamond_{[0,6e-5)} |y-5| \leq 3.24$$

is implied by the following safety formula: $\varphi_s = \square_{[0.001+2\tau, 0.0147-2\tau]\times A} |y-5| \leq 3.24$, so that verifying that $\mathcal{I}$ satisfies φ_s implies it also satisfies $[\varphi]_{3e-5}$. We again used SpaceEx, confirming that $\mathcal{I}$ satisfies φ_s. □

Hybrid nonlinear: This is a 3D hybrid nonlinear system, with three modes and a 1D input signal. It is modified from [20]. In each mode, the dynamics of the nominal model $\mathcal{M}$ are given by:

$$\mathcal{M} \begin{cases} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} = \begin{bmatrix} -(1+\gamma x_2^2)x_1 + 0.1u \\ -0.5(1-\gamma x_1^2)x_2 + 2x_3 \\ -(1-\gamma x_1)2x_2 - 0.5x_3 + 0.4u \end{bmatrix} \\ \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \gamma x_1 + x_2 \\ x_3 \end{bmatrix} \end{cases} \quad (4)$$

where γ is a mode-specific constant. The derived model $\mathcal{I}$ is obtained from $\mathcal{M}$ by linearizing the mode dynamics around the 0 equilibrium. In [20] it was established that with 0 input, the two systems' location-specific dynamics are approximately bisimilar. We ran the RRT method of Section IV-A on the two systems, using a uniform sampling distribution, Euclidian distance function, and a Model Predictive Controller for local motion planning to generate a tree with 1000 nodes. We computed the largest ε along all branches of the tree, which yielded a value of 7.157 for $\tau = 0.06$. To illustrate Thm. 2 for this case, we used S-TaLiRo to check that $\mathcal{I}$ satisfies φ:

$$\varphi = \diamond_{[0,4]\times[J_{MAX}]}(\square_{[0,0.4]}|y_1 - y_2| \leq 8)$$

where J_{MAX} is an upper bound on the number of jumps in [0,4]. S-TaLiRo reports no falsifying trajectories when trying to falsify the corresponding $[\varphi]_\tau$ for $\mathcal{M}$. □

VI. Related Work

In this paper we understand conformance as a notion that relates *systems*, as done in [38], rather than a system and its specification as in [14], [41]. The work in [38] studies conformance of embedded *software* using type systems and a notion of conformance that only relaxes time, whereas we are interested in hybrid system models of embedded cyber-*physical* systems with real-valued outputs and a relaxation of space as well time distances. The work in [30] provides an approximate method for verifying formal equivalence between a Simulink model and its corresponding C code; however it requires equality of outputs between the two (an extension is alluded to in the Conclusion), and does not account for timing differences. The approach to conformance of hybrid systems in [33] (building on [39]) results in untestable definitions, and falls in the domain of nondeterministic abstractions. Other approaches, like [9], require knowledge of the internal system structure, which is not necessary, in our case, for Def. 2.3.

We defined conformance via the $(T, J, (\tau, \varepsilon))$-closeness between hybrid trajectories, based on the work of Goebel and Teel [22]. A number of closeness measures between hybrid trajectories and systems exist. Measures based on bisimulation [19] and supnorms [10] only consider the differences in signal values at the same moment in time, which is not appropriate here since TS may have different domains. Other closeness measures, on the other hand, consider only differences in trajectories' timing, e.g., [24]. It can be shown that $(T, J, (\tau, \varepsilon))$-closeness provides a continuum of closeness degrees between the two extremes presented in [1]. The Skorokhod distance between trajectories used in [12] is related to $(T, J, (\tau, \varepsilon))$, but its use of bijective retimings is too restrictive in our context. More on the limitations of bijective retimings in the hybrid systems context can be read in [15, Section 5]. In the latter work, a generalization of $(T, J, (\tau, \varepsilon))$-closeness is presented as a pseudo-metric, but no computational procedure is given for computing this more general pseudo-metric.

The works closest to ours are [26] and [35]. In [26], (τ, ε)-bisimulation relations between metric transition systems are defined. The goal is to define robust approximate synchronization between systems (rather than conformance testing). $(T, J, (\tau, \varepsilon))$-closeness extends (τ, ε)-bisimulation relations in a straighforward manner to hybrid time domains, and we place it in a computational framework where the objective is to estimate the conformance degree. Later, Quesel [35] defined the notion of (τ, ε)-***similar*** traces, and proved a property transfer result between (τ, ε)-similar traces, which is a special case of our Thm. 1. The main differences with our work are three. First, unlike (τ, ε)-closeness, (τ, ε)-similarity requires the retiming relation to be order-preserving [35, Def. 17]. Whether this is important depends on the intended application. Allowing local 'disorder' might be necessary to deal with noisy signals as shown in Section II-C. Second, multiple jumps within the same time step are 'deleted' [35, Section 3.1] and not captured in (τ, ε)-similarity: this can be problematic in a number of applications where such events are indicative of bugs (e.g. race conditions in mixed-signal circuit verification, gear slippage in automotive applications, Zeno behavior arising out of high-level modeling [42], or code generation scenarios like [5]). Finally, our Thm. 1 generalizes the result in [35] to hybrid time domains, which allows us to explicitly take into account discrete events that are of interest to the designer, whereas they are ignored in [35]. It is also a generalization to non-order preserving retimings.

VII. Conclusions

In this paper, we have defined conformance between a system model and a system derived from it, by a process of simplification or implementation, as a degree of closeness between the outputs of the two systems. We then demonstrated two methods to approximate this degree of conformance. In future work, we plan on conducting a systematic comparison of the three optimization methods: Simulated Annealing, RRTs and multi-start KNITRO. We will consider coverage-based methods for guiding the RRT optimization and the choice of sampling distribution [14], and how to 'pre-design' a model such that the derived implementation satisfies certain desired properties. Finally we will apply this framework to concrete examples of derivation processes such as code generation, and illustrate how it helps debugging the derivation process.

Acknowledgment

This work was partially supported by NSF grant CNS 1350420.

APPENDIX

Proof: (Theorem 1)

The proof is by induction on the formula structure. We present only the cases for the positive predicate, the Until and the Release since the other cases are immediate. For convenience, given $i \in \mathbf{dom}(\mu)$, let $\rho'(i) = \{i' \in \mathbf{dom}(\mu') \mid |t(i) - t'(i')| \leq \tau, j(i) = j'(i'), \text{ and } \|y(i) - y'(i')\| \leq \varepsilon\}$ be the set of 'matching indices' for $\mu(i)$. By definition of achievable closeness (Def. 2.2), $\rho'(i)$ is non-empty for all i (note the use of nonstrict inequalities in its definition). Define the function $d : Y \times \mathcal{P}(Y) \to \mathbb{R}_+$ to be $d(y, S) = \inf_{s \in S} \|y - s\|$ if $y \notin S$ and $d(y, S) = \inf_{s \in \partial S} \|y - s\|$ if $y \in S$ where ∂S is the boundary of S.

Base Case: $(\mu, i) \models p$ implies that $\mathbf{pr}_1(\mu(i)) \in \mathcal{O}(p)$. Now because $\mu \approx_{(\tau,\varepsilon)} \mu'$, $\rho'(i)$ is non-empty and for all $i' \in \rho'(i)$, $\|y(i) - y'(i')\| \leq \varepsilon$. By the triangle inequality, for any point $w \in \mathcal{O}(p)$, $\|y(i) - w\| + \|w - y'(i')\| \leq \|y(i) - y'(i')\|$. Taking the infimum over w twice on the left hand side, it comes that $d(y'(i'), \mathcal{O}(p)) \leq \varepsilon - d(y(i), \mathcal{O}(p)) \leq \varepsilon$. This implies $\mathbf{pr}_1(\mu'(i')) \in E(\mathcal{O}(p), \varepsilon)$ by definition of the expansion. So for all $i' \in \rho'(i)$, $\mathbf{pr}_1(\mu'(i')) \in \mathcal{O}_\varepsilon(p^+)$ and, thus, $(\mu', i') \models p^+ \equiv [p]_\tau$.

Until Case: Assume that $(\mu, i) \models_{\mathcal{O}} \varphi_1 \mathcal{U}_{\mathbb{I}} \varphi_2$. By definition there exists an integer i_2 such that $\mathbf{pr}_{2,3}(\mu(i_2)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I}$ and $(\mu, i_2) \models_{\mathcal{O}} \varphi_2$ and for all i_1 such that $i \leq i_1 < i_2$ we have $(\mu, i_1) \models_{\mathcal{O}} \varphi_1$. Let i_2 be the smallest such integer. By the induction hypothesis, $\forall i_2' \in \rho'(i_2)$, $(\mu', i_2') \models_{\mathcal{O}_\varepsilon} [\varphi_2]_\tau$. Also by the induction hypothesis,

$$\forall i' \in \rho'(i),\ (\mu', i') \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau \tag{5}$$

We now need to see what happens between such i' and i_2'.

Since $\mathbf{pr}_{2,3}(\mu(i_2)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I}$, we have $\mathbf{pr}_{2,3}(\mu'(i_2')) \in \mathbf{pr}_{2,3}(\mu'(i')) \oplus \mathbb{I}_{\langle -2\tau, 2\tau\rangle}$. Let i_2' be the smallest such integer.

Now consider any $i' \in \rho'(i)$. For all i_1' such that $i' \leq i_1' < i_2'$, we ask the question: is $\Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau$ satisfied at i_1'? By the fact that $\mu \approx_{(\tau,\varepsilon)} \mu'$, there is some i_1 such that $i_1' \in \rho'(i_1)$. If it were always the case that $i \leq i_1 < i_2$, then the induction hypothesis would allow us to answer in the affirmative. But this is not always the case. Therefore, we need to consider three cases. Define i_2'' to be the smallest integer in $\mathbf{dom}(\mu')$ such that $t'(i_2'') > t'(i_2') - 2\tau$.

1) Case $i \leq i_1 < i_2$: then $(\mu, i_1) \models_{\mathcal{O}} \varphi_1$ by definition of the Until, and by the induction hypothesis, $(\mu', i_1') \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau$. A fortiori, $(\mu', i_1') \models_{\mathcal{O}_\varepsilon} \Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau$.
2) Case $i_1 < i$: In this case, we have no guarantee that $(\mu, i_1) \models_{\mathcal{O}} \varphi_1$, so the transformed formula needs to be made more permissive. That $i_1 < i$ and $i_1' \geq i'$ implies that $t'(i_1') < t'(i') + 2\tau$, as shown (see also Fig. 7, top):
$$\begin{aligned} t'(i_1') &< t(i_1) + \tau < t(i) + \tau \\ t'(i') > t(i) - \tau &\implies -t'(i') < -t(i) + \tau \\ &\implies t'(i_1') - t'(i') < 2\tau \\ &\implies t'(i_1') < 2\tau + t'(i') \end{aligned} \tag{6}$$
Moreover, $j(i_1) \leq j(i) = j'(i') \leq j'(i_1')$. But $j(i_1) = j'(i_1')$, so we have equalities throughout, and in particular
$$j'(i') = j'(i_1') \tag{7}$$
Therefore, by (5), (6) and (7),
$$(\mu', i_1') \models_{\mathcal{O}_\varepsilon} \Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau$$
3) Case $i_1 \geq i_2$: in this case there isn't much that can be said about the satisfaction of $[\varphi_1]_\tau$. However, we will make an argument similar to that of Case 2: along with $i_1' < i_2'$, this case implies that
$$t'(i_2') - t'(i_1') < 2\tau \tag{8}$$
as shown (see also Fig. 7 (bottom) for an illustration of these relations):
$$\begin{aligned} t'(i_2') &< t(i_2) + \tau \\ t'(i_1') &> t(i_1) - \tau \geq t(i_2) - \tau \\ &\implies t'(i_1') > t'(i_2') - 2\tau \end{aligned}$$
Moreover, $j(i_2) \leq j(i_1) = j'(i_1') \leq j'(i_2')$. But $j(i_2) = j'(i_2')$, so we have equalities throughout, in particular
$$j'(i_1') = j'(i_2') \tag{9}$$
This has two consequences: first, $\mathbf{pr}_{2,3}(\mu(i_1')) \in \mathbf{pr}_{2,3}(\mu(i')) \oplus \mathbb{I}_{\langle -2\tau, 2\tau\rangle}$. Second,
$$(\mu', i_1') \models_{\mathcal{O}_\varepsilon} \Diamond_{[0,2\tau)\times\{0\}}[\varphi_2]_\tau$$
Recall we defined i_2'' as the smallest integer in $\mathbf{dom}(\mu')$ such that $t'(i_2'') > t'(i_2') - 2\tau$ (it could be i_1' itself). And note that $\forall k' < i_2'', \exists\ k$ s.t. $k' \in \rho'(k)$ and $t'(k') \leq t'(i_2') - 2\tau$ and so $k < i_2$, so k' fits one of the first 2 cases. Then $(\mu', i_2'') \models_{\mathcal{O}_\varepsilon} \Diamond_{[0,2\tau)\times\{0\}}[\varphi_2]_\tau$, and $\mathbf{pr}_{2,3}(\mu(i_2'')) \in \mathbf{pr}_{2,3}(\mu'(i')) \oplus \mathbb{I}_{\langle -2\tau, 2\tau\rangle}$.

Combining the three cases, for all $i' \leq i_1' < i_2''$ (implying that $i_1 < i_2$), it holds that $\Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau$ is true of (μ', i_1').

In conclusion we may assert that $\forall i' \in \rho'(i)$,

$$(\mu', i') \models_{\mathcal{O}_\varepsilon} (\Diamond_{(-2\tau,0]\times\{0\}}[\varphi_1]_\tau) \mathcal{U}_{\mathbb{I}_{\langle -2\tau,2\tau\rangle}} (\Diamond_{[0,2\tau)\times\{0\}}[\varphi_2]_\tau)$$

Case 1 always holds if the $(T, J, (\tau, \varepsilon))$-closeness relation were order-preserving for a given pair (μ, μ'). In this case we can make the stronger assertion that for all $i' \in \rho'(i)$, $(\mu', i') \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau \mathcal{U}_{\mathbb{I}_{\langle -2\tau,2\tau\rangle}} [\varphi_2]_\tau$.

Release case: Assume that $(\mu, i) \models_{\mathcal{O}} \varphi_1 \mathcal{R}_{\mathbb{I}} \varphi_2$. let i' be any element of $\rho'(i)$. For any $k' \in \mathbf{dom}(\mu')$,

$$\begin{aligned} &\mathbf{pr}_{2,3}(\mu'(k')) \in \mathbf{pr}_{2,3}(\mu'(i')) \oplus \mathbb{I}_{\langle -2\tau,2\tau\rangle} \\ &\implies \mathbf{pr}_{2,3}(\mu'(k')) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I}_{\langle -\tau,\tau\rangle} \\ &\implies \exists k \in [N]. k' \in \rho'(k) \text{ and } \mathbf{pr}_{2,3}(\mu(k)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I} \end{aligned}$$

By definition of the Release it holds either that
(A) $(\mu, k) \models_{\mathcal{O}} \varphi_2$, or
(B) $\exists i \leq \ell < k. (\mu, \ell) \models_{\mathcal{O}} \varphi_1$.
If (A), then by the induction hypothesis $\forall k'' \in \rho'(k)$, $(\mu', k'') \models_{\mathcal{O}_\varepsilon} [\varphi_2]_\tau$, in particular at $k'' = k'$. If (B), then without loss of generality, assume $\rho'(\ell) = \{\ell'\}$. We distinguish three cases:

(B.1) $i' \leq \ell' < k'$: then by the induction hypothesis, $(\mu', \ell') \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau$.

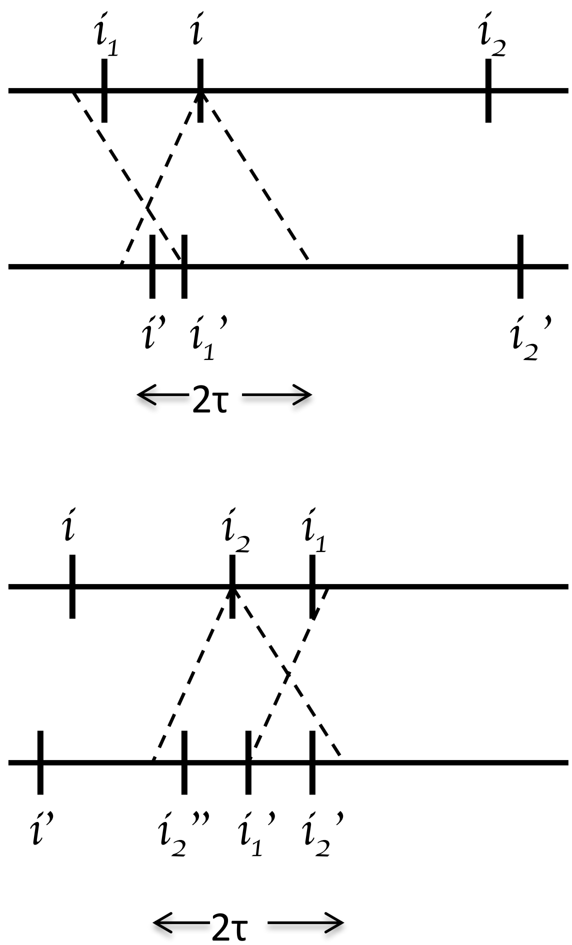

Fig. 7: Until case. Top: Case 2. Bottom: Case 3. In this and all figures of the appendix, the horizonal lines represent real time, and the indices mark the location of real time, i.e. i is located at $t(i)$, etc.

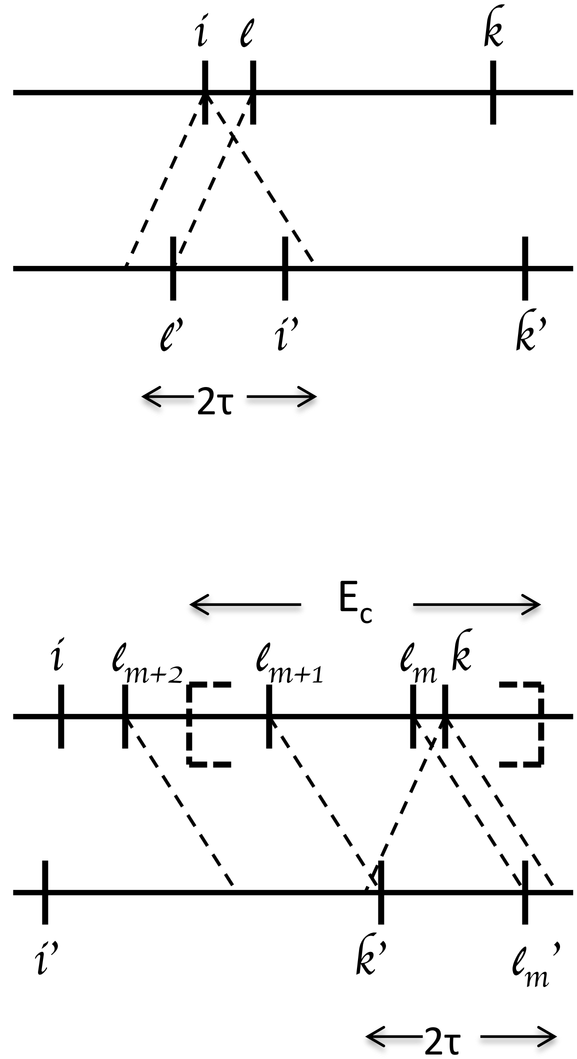

Fig. 8: Release case. Top: Case B.2. Bottom: Case B.3.

(B.2) $\ell' < i'$: combined with $\ell \geq i$, this implies that $t'(\ell') > t'(i') - 2\tau$ and that $j'(\ell') = j'(i')$ - this can be shown along the same lines as (6) and (7) of the Until case. See also Fig. 8 (top). Therefore $(\mu', i') \models_{\mathcal{O}_\varepsilon} \Diamond_{(-2\tau,0]}[\varphi_1]_\tau$.

(B.3) $\ell' \geq k'$: combined with $\ell < k$, this implies that $t'(k') \leq t'(\ell') < t'(k') + 2\tau$. See Fig. 8 (bottom) for an illustration. Therefore $t'(k') - \tau < t(\ell) \implies t(\ell) \in E_c \implies \mathbf{pr}_{2,3}(\mu(\ell)) \in \mathbf{pr}_{2,3}(\mu(i)) \oplus \mathbb{I}$. We will reason iteratively: set $\ell_0 = k$ (initialization), $m = 1, \ell_m = \ell$. Because $t(\ell_m) \in E_c$ and $\ell'_m \geq \ell'_k$,

- (C.1) either $(\mu, \ell_m) \models_{\mathcal{O}} \varphi_2$, in which case $(\mu', \ell'_k) \models_{\mathcal{O}_\varepsilon} \Diamond_{[0,2\tau)}[\varphi_2]_\tau$ and we're done, or
- (C.2) $\exists \ell_{m+1} < \ell_m$ s.t. $(\mu, \ell_{m+1}) \models_{\mathcal{O}} \varphi_1$.

In case (C.2), increment $m \leftarrow m+1$, and let $\ell'_m \in \rho'(\ell_m)$ (which we assume to be a singleton without loss of generality). Again,

- (C.2.1) either $\ell'_m < k'$ so $(\mu', \ell'_m) \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau$ and we're done.[1] Or,
- (C.2.2) $\ell'_m \geq k' \implies t(\ell_m) \in E_c$ and $t'(\ell_{m'}) < t'(k') + 2\tau$, and goto (C.1)

This iterations must stop at some point, either by (C.1), or by (C.2.1) because ℓ_m are strictly decreasing.

[1]Strictly speaking, we should handle the case $\ell'_m < i'$ separately, but this is handled exactly the same way as (B.2).

Combining cases (A) and (C.1) together, and cases (B.1) , (B.2) and (C.2) together, it comes that $\forall i' \in \rho'(i)$

$$(\mu', i') \models_{\mathcal{O}_\varepsilon} (\Diamond_{(-2\tau,0]}[\varphi_1]_\tau)\mathcal{R}_{\mathbb{I}_{\langle -2\tau, 2\tau\rangle}}(\Diamond_{[0,2\tau)}[\varphi_2]_\tau)$$

■

Note that the 'eventually' terms arise because of local disorder at the 'boundaries' of the formula, i.e. at i and k for the Release case, and at i and i_2 for the Until case. The statement can be strengthened if we restrict ourselves to consider the satisfaction away from those boundaries.

We now give an example to show that the above results are the best achievable without any further assumptions.

Example 3: Consider the situation in Fig. 9, where the top axis represents $\mathbf{pr}_2(\mu)$ and the bottom axis represents

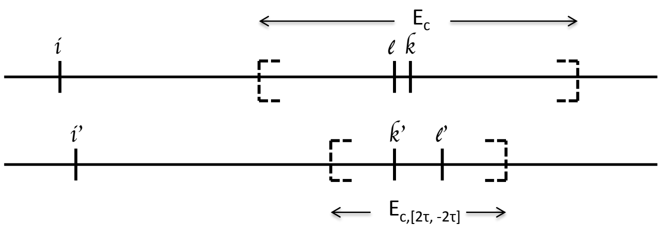

Fig. 9: Example 3

$\mathbf{pr}_2(\mu')$. Set $\mathbb{I} = E_c \times E_d$ for some E_d. We are given that $(\mu, i) \models_{\mathcal{O}} \varphi_1 \mathcal{R}_{\mathbb{I}} \varphi_2$, and that $(\mu, k) \not\models_{\mathcal{O}} \varphi_1 \vee \varphi_2$. Assume that $i' \in \rho'(i), k' \in \rho'(k)$, $\ell' \in \rho'(\ell)$. This implies $(\mu, \ell) \models_{\mathcal{O}} \varphi_1 \wedge \varphi_2$. We know that $(\mu, \ell') \models_{\mathcal{O}_\varepsilon} [\varphi_1]_\tau$. The most we can say about (μ', k') is that $(\mu', k') \models \Diamond_{[0,2\tau)}[\varphi_2]_\tau$ □

Proof: (Theorem 2)

By definition of simulation, every output TS $\mu_1 = (y_1, t_1, j_1)$ of $\mathcal{H}_1$ has a corresponding TS $\mu_2 = (y_2, t_2, j_2)$ of $\mathcal{H}_2$ that is $(T, J, (\tau, \varepsilon))$-close to it. By hypothesis, $(\mu_2, 0) \models_{\mathcal{O}} \varphi$. Define the mapping $\rho_1 : \mathbf{dom}(\mu_2) \to \mathbf{dom}(\mu_1)$ by $\rho_1(i') = \{i \in \mathbf{dom}(\mu_1) \mid |t_1(i) - t_2(i')| \leq \tau, j_1(i) = j_2(i'), \|y_1(i) - y_2(i')\| \leq \varepsilon\}$. It follows from Thm. 1 that $(\mu_1, i) \models_{\mathcal{O}_\varepsilon} [\varphi]_\tau$ for all $i \in \rho_1(0)$. Since $t_2(0) = 0$, it comes that $t_1(i) \leq \tau$ for all $i \in \rho_1(0)$. ■

Proof: (Proposition 4.1)

Given $z = (\mu_1, \mu_2) \in \mathcal{Z}$ with $\mathbf{dom}(\mu_1) = [N]$, and $i \in [N]$, define

$$W(i, z) := \{k \in S(i) \mid j_1(i) = j_2(i+k), |t_1(i) - t_2(i+k)| < \tau\}$$

$$f(k, i, z) := \|y_1(i) - y_2(i+k)\|^2$$

The set $S(i, z)$ contains all indices k such that $f(k, i, z)$ is well-defined, and the set S, if non-empty, contains all indices k such that g_k is well-defined.

Given $k \in S$, define the following indices

$$\begin{aligned} k^*(i, z) &= \arg \min_{k \in W(i,z)} f(k, i, z) \\ i_0(z) &= \arg \max_{i \in [N]} f(k^*(i, z), i, z) \\ i^*(k, z) &= \arg \max_{i \in [N]} f(k, i, z) \\ k_0(z) &= \arg \min_{k \in W(i_0, z)} f(k, i^*(k, z), z) \end{aligned}$$

Now for any i, z, k, $f(k^*(i, z), i, z) \leq f(k, i^*(k, z), z)$, in particular, at i_0, k_0, $f(k^*(i_0, z), i_0, z) \leq f(k_0, i^*(k_0, z), z)$, i.e.

$$\forall z \in \mathcal{Z}, \underbrace{\max_{i \in [N]} \min_{k \in W(i,z)} f(k, i, z)}_{LHS} \leq \min_{k \in W(i_0,z)} \underbrace{\max_{i \in [N]} f(k, i, z)}_{g_k(z)}$$

$$(\varepsilon_1^*)^2 = \sup_z LHS \leq \sup_z \min_{k \in W(i_0,z)} g_k(z)$$

We further upper bound the right-hand side of the preceding inequality:

$$\begin{aligned} \forall z \in \mathcal{Z}, k \in S, \quad & \min_{k' \in W(i_0,z)} g_{k'}(z) \leq \sup_{z'} g_k(z') \\ \implies \forall k, \quad & \sup_z \min_{k' \in W(i_0,z)} g_{k'}(z) \leq \sup_{z'} g_k(z') \\ \implies \quad & \sup_z \min_{k' \in W(i_0,z)} g_{k'}(z) \leq \min_{k \in S} \sup_z g_k(z) \\ \implies \quad & \varepsilon_1^* \leq \sqrt{\min_{k \in S} \sup_z g_k(z)} \end{aligned}$$

Note that if $S = \{0\}$, then the upper bound reduces to the supnorm: $g_0(z) = \max_{i \in [N]} \|y_1(i) - y_2(i)\|^2$. Note also that the fact that the supnorm upper bounds ε_1^* holds more generally than in the setting of Section IV-B. ■

Proof: (Proposition 4.2) The fact that the integration (or sampling) step is fixed, and that both systems are subject to the same external switching signal, imply that $\mathbf{dom}(\mu) = \mathbf{dom}(\mu') = [N]$ for all output TS of the two systems. Thus the hypothesis of Prop. 4.1 is satisfied since $0 \in S$, and g_k is well-defined.

To prove convexity, note that g_k is the maximum of $N+1$ functions of z, each of the form $\|y_1(i) - y_2(i+k)\|^2$. The norm squared $\|\cdot\|^2$ is convex in its argument; $y_2(i)$ and $y_1(i+k)$ are linear in $\eta_2(i)$ and $\eta_1(i+k)$, respectively; $\eta_2(i)$ and $\eta_1(i+k)$ are obtained by simply projecting z onto the appropriate subspace of $\mathcal{Z}$, which is a linear operation. Therefore, g_k is convex in z for every k. ■

REFERENCES

[1] A. Abate and M. Prandini. Approximate abstractions of stochastic systems: A randomized method. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 4861–4866, 2011.

[2] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Conformance testing as falsification for cyber-physical systems. Technical Report arXiv:1401.5200, January 2014.

[3] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Work in progress: Conformance testing as falsification for cyber-physical systems. In *Cyber-Physical Systems, 2014 IEEE Intl. Conference on*, April 2014.

[4] H. Abbas, H. Mittelmann, and G. Fainekos. Formal property verification in a conformance testing framework. [Online at: http://www.public.asu.edu/~hyabbas/techreports/MEMOCODE14TechRpt.pdf], 2014.

[5] M. Anand, S. Fischmeister, Y. Hur, J. Kim, and I. Lee. Generating reliable code from hybrid-systems models. *Computers, IEEE Transactions on*, 59(9):1281–1294, Sept 2010.

[6] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[7] S. Bensalem, A. Legay, and M. Bozga. Rigorous embedded design: challenges and perspectives. *STTT*, 15(3):149–154, 2013.

[8] N. Bombieri, F. Fummi, G. Pravadelli, and J. Marques-Silva. Towards equivalence checking between TLM and RTL models. In *Proceedings of the 5th IEEE/ACM MEMOCODE*, pages 113–122, Washington, DC, USA, 2007.

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

[10] S. Burden, H. Gonzales, R. Vasudevan, R. Bajcsy, and S. S. Sastry. Metrization and simulation of controlled hybrid systems. Technical Report arXiv:1302.4402, February 2013.

[11] K. Butts. Presentation: Toyota's direction. [Online at: http://cmacs.cs.cmu.edu/presentations/verif_csystems/06_KenButts.pdf], 2010.

[12] P. Caspi and A. Benveniste. Toward an approximation theory for computerized control. In *Embedded Software*, volume 2491 of *LNCS*, pages 294–304. Springer, 2002.

[13] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.

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

[15] J. Davoren. Epsilon-tubes and generalized skorokhod metrics for hybrid paths spaces. In R. Majumdar and P. Tabuada, editors, *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 135–149. Springer Berlin Heidelberg, 2009.

[16] G. Di Guglielmo, M. Fujita, F. Fummi, G. Pravadelli, and S. Soffia. EFSM-based model-driven approach to concolic testing of system-level design. In *9th IEEE/ACM MEMOCODE*, pages 201–209, July 2011.

[17] G. Frehse, C. L. Guernic, A. Donz, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*, 2011.

[18] G. Funchal and M. Moy. Modeling of time in discrete-event simulation of systems-on-chip. In *9th IEEE/ACM MEMOCODE*, pages 171–180, July 2011.

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

[20] A. Girard and G. J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference*, pages 684–689, 2005.

[21] R. Goebel, R. G. SanFelice, and A. R. Teel. *Hybrid Dynamical Systems: modeling, stability and robustness*. Princeton University Press, 2012.

[22] R. Goebel and A. Teel. Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica*, 42(4):573 – 587, 2006.

[23] R. J. Hanson and D. C. Sorensen. Model reduction of dynamical systems for real time control. [Online at: http://www.caam.rice.edu/~modelreduction/mission.html].

[24] J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in approximations of timed systems. In *Formal Methods and Models for Co-Design, 2003. MEMOCODE '03. Proceedings. First ACM and IEEE International Conference on*, pages 163–171, June 2003.

[25] K. H. Johansson, J. Lygeros, S. Sastry, and M. Egerstedt. Simulation of hybrid zeno automata. In *Conference on Decision and Control*, volume 4, pages 3538–3543, December 1999.

[26] A. Julius and G. Pappas. Approximate equivalence and approximate synchronization of metric transition systems. In *Decision and Control, 2006 45th IEEE Conference on*, pages 905–910, Dec 2006.

[27] A. A. Julius, G. Fainekos, M. Anand, I. Lee, and G. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 329–342. Springer-Verlag Berlin Heidelberg, 2007.

[28] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *I. J. Robotic Res.*, 30(7):846–894, 2011.

[29] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[30] R. Majumdar, I. Saha, K. Ueda, and H. Yazarel. Compositional equivalence checking for models and code of control systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1564–1571, Dec 2013.

[31] R. Majumdar, I. Saha, and Z. Wang. Systematic testing for control applications. In *8th IEEE/ACM MEMOCODE*, pages 1–10, July 2010.

[32] L. V. Nguyen and T. J. Johnson. Benchmark: DC-to-DC switched-mode power converters (buck converters, boost converters and buck-boost converters). In *ARCH 2014*. 2014.

[33] M. Osch. Hybrid input-output conformance and test generation. In K. Havelund, M. Nez, G. Rou, and B. Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification*, volume 4262 of *Lecture Notes in Computer Science*, pages 70–84. Springer Berlin Heidelberg, 2006.

[34] A. Platzer and J.-D. Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In A. Armando, P. Baumgartner, and G. Dowek, editors, *International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[35] J.-D. Quesel. *Similarity, Logic, and Games: Bridging Modeling Layers of Hybrid Systems*. PhD thesis, Carl Von Ossietzky Universitat Oldenburg, July 2013.

[36] A. Saadat. Defect information report. [Online at: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.PDF], 2014.

[37] R. G. Sanfelice. Interconnections of hybrid systems: Some challenges and recent results. *Journal of Nonlinear Systems and Applications*, 2(1-2):111–121, 2011.

[38] J.-P. Talpin, P. Guernic, S. Shukla, and R. Gupta. A compositional behavioral modeling framework for embedded system design and conformance checking. *International Journal of Parallel Programming*, 33(6):613–643, 2005.

[39] J. Tretmans. Testing concurrent systems: A formal approach. In *CONCUR 1999 Concurrency Theory*, pages 46–65. Springer, 1999.

[40] Y. Watanabe and S. Swan. Clearing the clutter: Unified modeling and verification methodology for system level hardware design. In *10th IEEE/ACM MEMOCODE*, pages 21–23, July 2012.

[41] M. Woehrle, K. Lampka, and L. Thiele. Conformance testing for cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 11(4):84:1–84:23, Jan. 2013.

[42] J. Zhang, K. Johansson, J. Lygeros, and S. Sastry. Dynamical systems revisited: Hybrid systems with zeno executions. In N. Lynch and B. Krogh, editors, *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 451–464. Springer Berlin Heidelberg, 2000.

[43] Ziena. KNITRO. [Online at: http://www.ziena.com].