Hybrid PALS for Distributed Cyber-Physical Systems and Their SMT-Based Analysis

Kyungmin Bae, Soonho Kong
Carnegie Mellon University

Sicun Gao
MIT CSAIL

Peter Csaba Ölveczky
University of Oslo

Edmund M. Clarke
Carnegie Mellon University

***Abstract*—The PALS methodology reduces the problem of designing and verifying a *virtually synchronous* distributed real-time system to the much simpler task of designing and verifying its underlying *synchronous* design. This paper presents *Hybrid PALS*, which extends PALS to the important class of virtually synchronous cyber-physical systems (CPSs) where distributed controllers communicate with each other and interact with physical entities having *continuous dynamics*. We prove a bisimulation equivalence between the synchronous and the distributed models. We then explain how a number of verification problems for synchronous Hybrid PALS models can be reduced to SMT solving over real numbers with nonlinear ordinary differential equations with arbitrary precision. Since such SMT-based analysis typically becomes unfeasible due to the *formula explosion problem*, we propose a new SMT framework to effectively encode distributed hybrid systems in a modular way. We illustrate the effectiveness of the Hybrid PALS design and verification methodology on a number of CPSs, including a system for turning an airplane.**

I. Introduction

Virtually synchronous distributed real-time systems consist of a number of *distributed* controllers that should logically behave in a synchronous way. [[The devices may operate at different frequencies, but communication happens at their hyper-period boundary.]] Designing and analyzing such systems is very hard because of race conditions, clock skews, network delays, execution times, and the state space explosion caused by the interleavings. To overcome these problems, the *PALS* (physically asynchronous, logically synchronous) methodology [1]–[4] has been developed to reduce the design and verification of a virtually synchronous distributed real-time system to the much simpler task of designing and verifying the underlying *synchronous* models, provided that the infrastructure can guarantee bounds on computation times, network delays, and imprecision of the local clocks.

In this paper we present *Hybrid PALS* that extends PALS to the important class of virtually synchronous *cyber-physical systems* (CPSs)—where controllers also interact with their local *physical environments*, whose *continuous dynamics* can be specified by ordinary differential equations (ODEs)—which includes avionics, automotive, robotics, and medical systems. We prove a bisimulation equivalence relating distributed hybrid systems and their underlying synchronous models. The initial steps towards a hybrid extension of PALS were taken in [5]. However, that work imposes strong conditions on sampling and response times of sensors and actuators, and, furthermore, does not give a bisimulation equivalence.

Verifying the synchronous Hybrid PALS models is still challenging: inputs and parameter values can be any real number in certain intervals; different components read their physical values and give actuator commands at different times because of local clock skews, and these timings cannot be abstracted away; and the continuous dynamics often involves *nonlinear* ODEs which in general may not have exact solutions.

This paper presents SMT solving techniques to address the challenges of analyzing Hybrid PALS models of CPSs. The verification of a distributed CPS involving ODEs and clock skews is reduced to checking the satisfiability of SMT formulas over real numbers and ODEs, which is decidable up to any user-given precision $\delta > 0$ [6], [7].

One problem with SMT-based methods is that they do not scale well for distributed hybrid systems with nonlinear ODEs. We cope with this scalability problem as follows. First, the discrete part and the continuous part of the system are encoded separately, so that ODEs are considered *only after* the discrete part of the system has been fully analyzed. Second, in addition to bounded reachability analysis, *inductive* SMT analysis is used to verify safety properties, without any time bound. Third, compositional SMT analysis is used to analyze each part of a distributed CPS by a divide-and-conquer approach.

In distributed CPS, the physical states of different components can be correlated to each other. Consider, for example, two adjacent rooms: the temperature of one room then immediately affects the temperature of the other room. Such behaviors are specified as *coupled* ODEs where variables evolve simultaneously over time. Existing SMT approaches use non-modular encodings for such distributed hybrid systems. The size of the SMT formula can therefore be huge, which leads to the *formula explosion problem* that makes SMT-based analysis practically infeasible.

To overcome this problem, we present a new SMT framework to effectively encode formal analysis problems of distributed hybrid systems with coupled ODEs *in a modular way*. The key idea is to use equation "names" to logically decompose coupled variables in systems of ODEs in the SMT formula. We show that a satisfaction problem in the new theory can be reduced to one in the standard theory of the real numbers *at no cost*. We have implemented our SMT techniques within the dReal SMT solver [8]. We have applied our techniques on a range of advanced CPSs, including: a system for turning an airplane, networked water tank controllers, and networked thermostat controllers. The experimental results

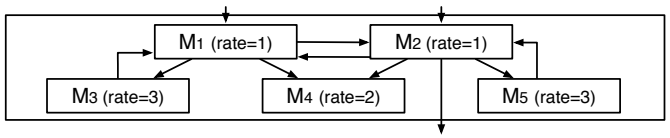

Fig. 1. A multirate ensemble $\mathcal{E}$, with M_1 and M_2 slow machines.

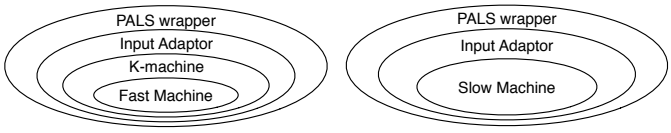

Fig. 2. The wrapper hierarchies in PALS distributed real-time models.

show that these techniques can dramatically improve the performance of SMT-based verification of distributed CPSs.

The rest of the paper is organized as follows. Section II gives a background on PALS. Section III introduces Hybrid PALS. Section IV shows how synchronous Hybrid PALS models and their verification problems (such as bounded reachability and inductive and compositional analysis) can be symbolically encoded as logical formulas. Section V presents a new SMT framework supporting the modular encoding of distributed hybrid systems. Section VI gives an overview of the Hybrid PALS verification case studies. Finally, Section VII discusses related work, and Section VIII gives some concluding remarks.

II. Preliminaries on PALS

PALS transforms a *synchronous design* SD with period T into a distributed real-time system $\mathcal{MA}(SD, T, \Gamma)$ that satisfies the same temporal logic (CTL^*) formulas, provided that the underlying infrastructure guarantees bounds $\Gamma = (\epsilon, \alpha_{\min}, \alpha_{\max}, \mu_{\min}, \mu_{\max})$ with (i) ϵ a maximal clock skew with respect to the global clock, (ii) $[\alpha_{\min}, \alpha_{\max}]$ bounds for processing I/O and executing a transition, and (iii) $[\mu_{\min}, \mu_{\max}]$ bounds for the network transmission delay. This section overviews the synchronous models, the distributed models, and their relationship (we refer to [2], [3] for details).

A. Discrete Synchronous Models

The synchronous model SD is specified as an *ensemble* $\mathcal{E}$ of state machines with input and output ports. In each iteration, a machine performs a transition based on its current state and its inputs, proceeds to the next state, and generates new outputs for the next iteration.

Definition 1. *A* typed machine $M = (D_i, S, D_o, \delta_M)$ *is composed of: (i)* $D_i = D_{i_1} \times \cdots \times D_{i_n}$ *an input set (a value to the* k*-th* input port *is an element of* D_{i_k}*), (ii)* S *a set of states, (iii)* $D_o = D_{o_1} \times \cdots \times D_{o_m}$ *an output set, and (iv)* $\delta_M \subseteq (D_i \times S) \times (S \times D_o)$ *a total transition relation.*

As illustrated in Fig. 1, a collection $\{M_j\}_{j \in J_S \cup J_F}$ of state machines with different periods can be composed into a *multirate ensemble* $\mathcal{E}$. The period of a slow machine $s \in J_S$ (with $rate(s) = 1$) is a multiple of the period of a fast machine $f \in J_F$ (with $rate(f) > 1$). A *wiring diagram*, which has no connections between two fast machines, connects the input ports and output ports of the machines.

In each iteration, all components in $\mathcal{E}$ perform a transition each *in lockstep*. A fast machine f is *slowed down* and performs $k = rate(f)$ *internal* transitions in one global synchronous step. Since a fast machine produces k-tuples of outputs in one step, *input adapters* are used to generate single values (e.g., the last value, or the average of the k values) for a slow machine. Likewise, a single output from a slow machine is adapted to a k-tuple of inputs for a fast machine.

The *synchronous composition* of a multirate ensemble $\mathcal{E}$ is equivalent to a single machine $M_{\mathcal{E}}$. If a machine in $\mathcal{E}$ has a feedback wire connected to itself or to another component, then the output becomes an input of the destination component in the next iteration. That is, $M_{\mathcal{E}}$'s states consist of the states of its subcomponents and the "feedback" outputs. For example, $M_{\mathcal{E}}$ of $\mathcal{E}$ in Fig. 1 is the machine given by the outer box.

The global *environment* can be modeled as another typed machine, and *local* environments are assumed to be composed with their controllers into a single typed machine. This model is an untimed model (apart from the period T): the result of applying a (controller or environment) transition is independent of *when* the transition is applied in a round.

B. PALS Distributed Real-Time Models

Each component in the distributed model $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ is composed of a machine in $\mathcal{E}$ and *wrappers* around it, as illustrated in Fig. 2. In $\mathcal{MA}(\mathcal{E}, T, \Gamma)$, each machine performs at its own rate according to its local clock. At the beginning of its periods, it reads its input from the layer above, performs a transition, and then generates the outputs.

A wrapper has I/O buffers, timers, and access to the machine's local clock. Each PALS wrapper has the same *global period* T and stores received inputs in its input buffer. When the i-th round begins (at time $u_0 \in (iT - \epsilon, iT + \epsilon)$), it delivers the contents of its input buffer to the inner input adaptor wrapper, and sets its *backoff timer* to $2\epsilon - \mu_{min}$. When the execution of the inner components is finished *and* the backoff timer expires, the contents of the output buffer are sent out.[1]

An input adaptor wrapper reads the inputs from the PALS wrapper and applies input adaptors for each global period T. A k-machine wrapper (i) extracts each value from the k-tuple input and delivers it to the enclosed fast machine at each fast period T/k, and (ii) delivers the k*-tuples* from the outputs of the fast machine to its outer layer at each global period T.

A fast machine M_f may *not* be able to finish all of its k internal transitions in a global round *before* the outputs must be sent to arrive before the next round. The number of transitions that M_f can perform before the deadline is $k' = 1 + \lfloor \max(T - (2\epsilon + \mu_{max} + \alpha_{max_f}), 0) \cdot (k/T) \rfloor$, where $\alpha_{\max_f}$ is the maximal execution time for M_f. If $k' < k$,

[1]The outputs are sent out before $u_0 + \max(2\epsilon - \mu_{min}, \alpha_{max})$, and delivered to the destination before $u = \mu_{max} + u_0 + \max(2\epsilon - \mu_{min}, \alpha_{max})$. Therefore, if $T \geq 2\epsilon + \mu_{max} + \max(2\epsilon - \mu_{min}, \alpha_{max})$, then all inputs are read in a round-consistent way since $u < (i+1)T - \epsilon$ [3].

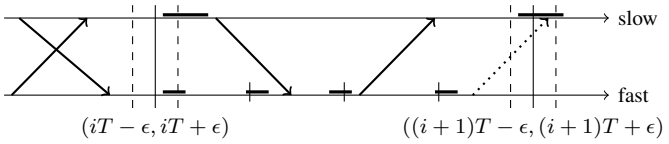

Fig. 3. Timeline for $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ ($k = 4$ and $k' = 3$). Diagonal arrows denote network transmission and short horizontal lines denote the execution. The dotted arrow illustrates that the outputs can arrive after the beginning of the next round if the fast machine waits until all its transitions are finished.

then M_f's k-machine wrapper only sends the first k' values (followed by $k-k'$ "don't care" values $\perp$). The input adaptor of each input port whose source is M_f must ignore the last $k-k'$ values $v_{k'+1}, \ldots, v_k$ in a k-tuple $(v_1, \ldots, v_k)$.

C. Relating the Synchronous and Distributed Models

In $\mathcal{MA}(\mathcal{E}, T, \Gamma)$, network transmission can happen only in the time interval $(iT+\epsilon, (i+1)T-\epsilon)$ for each i-th period, as depicted in Fig. 3. Therefore, at each time $iT-\epsilon$, all the input buffers of the PALS wrappers are full, and all the other input and output buffers are empty. A *stable state* of the distributed model $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ is a snapshot of the system at each time $iT-\epsilon$, just before the components in $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ start performing local machine transitions [3].

Big-step transitions are defined between two stable states of $\mathcal{MA}(\mathcal{E}, T, \Gamma)$, and they are related to single steps of the synchronous composition $M_{\mathcal{E}}$ through the function *sync* associating to each stable state the corresponding state in $M_{\mathcal{E}}$. Two stable states are related by $s_1 \sim_{obi} s_2$ iff their machine states are identical and their corresponding input buffer contents *cannot* be distinguished by input adaptors.

Theorem 1. *[2] The relation $(\sim_{obi}; sync)$ is a bisimulation between the transition system induced by $M_{\mathcal{E}}$ and the big-step stable transition system induced by $\mathcal{MA}(\mathcal{E}, T, \Gamma)$.*

III. HYBRID PALS

This section presents *Hybrid PALS*, which extends PALS to hybrid systems. One main difference between PALS and Hybrid PALS is that the time at which an event takes place does not matter in PALS, as long as it happens within a certain time interval. This allows us to relate a distributed real-time system to an essentially untimed synchronous model. However, in hybrid systems, we cannot abstract from the time at which a continuous value is read or an actuator command is given (both of which depend on a component's imprecise local clock). Therefore, the precise times at which these events happen must be included also in the *synchronous* Hybrid PALS models to have any possibility of an equivalence between the synchronous model and the distributed hybrid system.

In Hybrid PALS, a state of a physical environment is a tuple $\vec{v} = (v_1, \ldots, v_l) \in \mathbb{R}^l$ of its physical parameters $\vec{x} = (x_1, \ldots, x_l)$, and the behavior of $\vec{x}$ can be modeled by ODEs that specify *trajectories* $\tau_1, \ldots, \tau_l$ of $\vec{x}$ over time. The standard PALS models $\mathcal{E}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ are nondeterministic models defined for *any possible* environment behaviors.

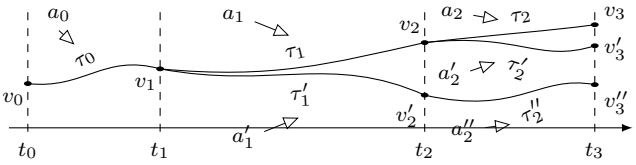

Fig. 4. A controlled physical environment E_M.

The *environment restrictions* $\mathcal{MA}(\mathcal{E}, T, \Gamma) \upharpoonright E$ and $\mathcal{E} \upharpoonright E$ define the behavior of the models constrained by the physical environment E. To have any control over when values are read from, and sent to, physical environments, we add in this paper *sampling and response timing policies* to the Hybrid PALS models. We then prove a bisimulation equivalence relating $\mathcal{E} \upharpoonright E$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \upharpoonright E$, which generalizes the trace equivalence result in [5].

A. Controlled Physical Environments

A (local) environment E_M of machine M is specified as a *controlled physical environment* that defines every possible trajectory of its physical parameters for the control commands from M. For a state $\vec{v} \in \mathbb{R}^l$, a control command a, and a duration $t \in \mathbb{R}$, a physical environment E_M gives a trajectory $\vec{\tau}$ of its parameters $\vec{x}$ of duration t, as illustrated in Fig. 4.

Definition 2. *Let $\mathcal{T}$ denote the set of all trajectories (a trajectory of duration T is a function $\tau : [0, T] \to \mathbb{R}$). A* controlled physical environment $E_M = (C, \vec{x}, \Lambda)$ *consists of: (i) C a set of* control commands*; (ii) $\vec{x} = (x_1, \ldots, x_l)$ a vector of real number variables; and (iii) $\Lambda \subseteq (C \times \mathbb{R}_{\geq 0} \times \mathbb{R}^l) \times \mathcal{T}^l$ a* physical transition relation*, where $((a, t, \vec{v}), \vec{\tau}) \in \Lambda$ iff for a control command $a \in C$ that lasts for duration t, E_M's physical state $\vec{x}$ follows the trajectory $\vec{\tau} \in \mathcal{T}^l$ from $\vec{v} \in \mathbb{R}^l$ with $\vec{\tau}(0) = \vec{v}$. (E.g., $((a_1, t_2 - t_1, v_1), \tau_1) \in \Lambda$ in Fig. 4).*

Several physical environments may be physically correlated, and one environment may affect another environment. Such correlations are expressed as *time-invariant constraints* $(\forall t.\, \psi)$ of physical parameters over time t. For example, if parameter x_1 of E_{M_1} must be equal to parameter x_2 of E_{M_2}, then the time-invariant constraint is $\forall t.\, x_1(t) = x_2(t)$.

B. Environment-Restricted Controllers

A controller M is a *nondeterministic* machine parameterized by any behavior of its environment E_M. The controller M interacts with E_M according to its local clock, which may differ from global time by up to the maximal clock skew ϵ. Let $c_M : \mathbb{N} \to \mathbb{R}_{>0}$ denote a a *periodic local clock* of M that gives the *global time* at the beginning of the $(i+1)$-th period according to M's local clock. That is, $c_M(0) = 0$ and $c_M(n) \in (nT-\epsilon, nT+\epsilon)$ for each $n > 0$.

Fig. 5 depicts the behavior of the *environment restriction* $M \upharpoonright E_M$ in a PALS distributed model. Its $(i+1)$-th period begins at time $c_M(i) \in (iT-\epsilon, iT+\epsilon)$. As explained in Section II-B, M has already received all the inputs $\vec{i}$ before time $iT-\epsilon$. At this time, M also samples the appropriate

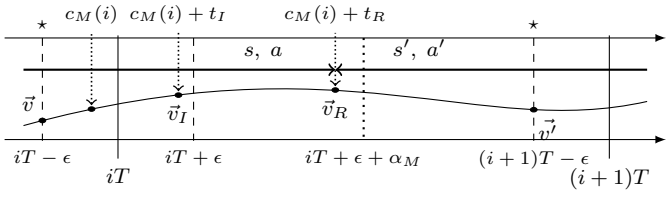

Fig. 5. Timeline for an environment-restricted controller $M \restriction E_M$ with a local clock c_M, an environment sampling time t_I, and an response time t_R.

values of its environment E_M. The sampling takes time t_I, so that the physical state $\vec{v}_I$ of E_M is read at time $c_M(i)+t_I$. M executes its transition based on the inputs $\vec{i}$, the physical state $\vec{v}_I$, and the current machine state s. After the execution ends, the machine state changes to s', and the new controller command a is sent to E_M at time $c_M(i)+t_R$, where t_R denotes the response time, which equals the transition execution time and the actuator processing time. The new outputs $\vec{o}$ from M are delivered to their destinations before time $(i+1)T-\epsilon$.

A control command from a controller M depends on M's current state. In Fig. 5, a current control command a remains effective until the execution ends at time $c_M(i)+t_R$, and then a new control command a' takes effect. A physical environment E_M defines trajectories of its parameters $\vec{x}$: (i) from state $\vec{v}$ to $\vec{v}_I$ for the interval $[iT-\epsilon, c_M(i)+t_I]$ by the current command a, (ii) from $\vec{v}_I$ to $\vec{v}_R$ for the interval $[c_M(i)+t_I, c_M(i)+t_R]$ by again a, and (iii) from $\vec{v}_R$ to $\vec{v'}$ for $[c_M(i)+t_R, (i+1)T-\epsilon]$ by the new control command a'. Notice that $0 \le t_I \le t_R \le \alpha_M$ for the maximal execution time α_M of M. As a result, a *big step* transition of $M \restriction E_M$ from state $(s, \vec{v})$ with input $\vec{i}$ to state $(s', \vec{v'})$ with output $\vec{o}$ is defined for the time interval $[iT-\epsilon, (i+1)T-\epsilon]$ in *distributed PALS models*.

A machine M that integrates both a controller and its environment should have a state space of the form $S \times \mathbb{R}^m$, where $\mathbb{R}^m$ is the state space of the m physical parameters that M observes. That M has a transition for *any* environment values follows directly from the definitions of machines: their transition relations are total. To compose such a machine with its environment, we must define the "interface" between the controller part and the "environment part" of such an environment-parametric machine M. This interface is given by the following projection functions:

Definition 3. *The* interface *of a controller M is given by the projection functions $\pi = (\pi_T, \pi_R, \pi_I, \pi_C, \pi_O)$, where $s \in S$ is a state in M: (i) $\pi_T(s) \in \mathbb{N}$ a "round" number; (ii) $\pi_R(s, \vec{i}) \in \mathbb{R}$ a response time; (iii) $\pi_I(s) \in \mathbb{R}$ a sampling time; and (iv) $\pi_C(s) \in C$ the control command of M to E_M. For state $\vec{v} \in \mathbb{R}^l$ of E_M, the* observable *part of $\vec{v}$ by M is given by the projection function $\pi_O(\vec{v}) \in \mathbb{R}^m$.*

The environment restriction of M can then be defined in the expected way:

Definition 4. *The* environment restriction *of M by E_M with an interface π is $M \restriction_\pi E_M = (D_i, S \times \mathbb{R}^l, D_o, \delta_{M \restriction_\pi E_M})$, where $((\vec{i}, (s, \vec{v})), ((s', \vec{v'}), \vec{o})) \in \delta_{M \restriction_\pi E_M}$ holds iff $\pi_T(s') = i+1$ for the round number $i = \pi_T(s)$, and:*

1) *$((a, u_I, \vec{v}), \tau_I) \in \Lambda$ and $\vec{v}_I = \tau_I(u_I)$, for $t_I = \pi_I(s)$, duration $u_I = (c_M(i)+t_I) - (iT-\epsilon)$, and the current controller command $a = \pi_C(s)$;*
2) *$((a, u_R - u_I, \vec{v}_I), \tau_R) \in \Lambda$ and $\vec{v}_R = \tau_R(u_R - u_I)$, for $t_R = \pi_R(s, \vec{i})$ and $u_R = (c_M(i)+t_R) - (iT-\epsilon)$;*
3) *$((\vec{i}, (s, \pi_O(\vec{v}_I))), ((s', \pi_O(\vec{v'})), \vec{o})) \in \delta_M$; and*
4) *$((a', T-u_R, \vec{v}_R), \tau_W) \in \Lambda$ and $\vec{v'} = \tau_W(T-u_R)$, for the next control command $a' = \pi_C(s')$.*

C. Hybrid PALS

Hybrid PALS models are parameterized by *sampling and response timing policies* that determine sensor sampling timing t_I and actuator response timing t_R. Such a policy is given as a collection of projection functions $\pi = (\pi_T, \pi_R, \pi_I, \pi_C, \pi_O)$ in Definition 3. The continuous dynamics is "completely" decided by the controllers, their physical environments, and timing policies (including local clocks of controllers). In [5], t_I is 0 (for controllers tightly integrated with their environments) and t_R is the maximum execution time $\alpha_{\max}$.

The synchronous model in Hybrid PALS is specified as a multirate ensemble $\mathcal{E}$ and the physical environments of subcomponents, where physical correlations between those environments are specified as time-invariant constraints.

Definition 5. *A* hybrid multirate ensemble *$\mathcal{E} \restriction_\Pi E_\mathcal{E}$ is given by: (i) a multirate ensemble $\mathcal{E}$, (ii) a family of policy functions $\Pi = \{\pi_j\}_{j \in J_S \cup J_F}$ (with J_S an index set of slow machines and J_F an index set of fast machines), and (iii) a family of local physical environments $E_\mathcal{E} = \langle \{E_{M_j}\}_{j \in J_S \cup J_F}, \forall t.\, \psi \rangle$ with $\forall t.\, \psi$ the time-invariant constraints.*

The behaviors of a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ are a *subset* of the behaviors of $\mathcal{E}$, namely, the behaviors restricted by the physical environment $E_\mathcal{E}$. As mentioned, each (decelerated) environment-restricted machine $M_j \restriction_\pi E_{M_j}$ defines a transition from a state at time $iT-\epsilon$ to one at time $(i+1)T-\epsilon$ for a global period T. Therefore, a lockstep composition of such transitions *that also satisfies the time-invariant constraints* gives a transition of the synchronous composition $M_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ from one state at time $iT-\epsilon$ to another state at time $(i+1)T-\epsilon$.

Hybrid PALS maps a hybrid multirate ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ with a global period T, together with PALS bounds Γ, to the distributed hybrid system $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$. Similarly, its behaviors are the subset of those of the distributed system $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ that can be realized by the environment $E_\mathcal{E}$ and satisfy the time-invariant constraints. In particular, big-step transitions of each component $M_j \restriction_\pi E_{M_j}$ are defined for the interval $[iT-\epsilon, (i+1)T-\epsilon]$ as explained in Section III-B.

The correctness of Hybrid PALS follows from the fact that all physical measurements and physical activation *happen at the same time* in both $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ with the *same timing policies* Π. Recall that stable states of $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ are states at times $iT-\epsilon$. Hence, big-step "stable" transitions are well defined for $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ from $iT-\epsilon$ to $(i+1)T-\epsilon$, which are exactly related to single

steps of the synchronous composition $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ (by the same relation $(\sim_{obi}; sync)$ in Theorem 1). Consequently, we have:

Theorem 2. *The relation $(\sim_{obi}; sync)$ is a bisimulation between the transition system by $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ and the big-step stable transition system induced by $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$.*

IV. Hybrid PALS Models as Logical Formulas

Theorem 2 implies that verifying a distributed hybrid CPS $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ amounts to verifying the much simpler synchronous model $\mathcal{E} \restriction_\Pi E_\mathcal{E}$. However, it is still challenging to analyze $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ with (nonlinear) continuous dynamics: The concrete values of the imprecise local clocks are unknown since they are determined on-the-fly by *clock synchronization* mechanisms, and numerical simulation cannot be used for nonlinear dynamics because numerical errors can be accumulated.

This section explains how verification problems for $\mathcal{E} \restriction_\Pi E_\mathcal{E}$, for *all possible local clocks*, can be encoded as logical formulas over the real numbers and ODEs. The satisfiability of such formulas can then be automatically decided by SMT solving up to a given precision $\delta > 0$ as shown in Section V.

A. Encoding Environment-Restricted Controllers

A controller M can be expressed as a logical formula $\phi_M(\vec{i}, \vec{y} \shortmid \vec{y'}, \vec{o})$, with variables $\vec{i}$, $\vec{y}$, $\vec{y'}$, and $\vec{o}$ denoting, respectively, input, the current state, the next state, and output.

Definition 6. *For $M = (D_i, S, D_o, \delta_M)$, the formula ϕ_M is defined as $\phi_M(\vec{d_i}, s \shortmid s', \vec{d_o}) \iff ((\vec{d_i}, s), (s', \vec{d_o})) \in \delta_M$.*

A physical environment E_M can be encoded as a formula $\phi_{E_M}(\vec{a}, u_0, u_t, \vec{v} \shortmid \vec{x})$ over the real numbers with: *unary function symbols* $\vec{x}$ denoting the trajectories of E_M's physical parameters $\vec{x}$, and *variables* $\vec{a}$, u_0, u_t, and $\vec{v}$ denoting, respectively, control commands, the times at the beginning and the end of the trajectory duration, and the initial values of the trajectories $\vec{x}$ at time u_0.

If the continuous dynamics of $\vec{x}$ is specified as a system of ODEs $\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} = F_{\vec{a}}(\vec{x}, t)$ for a control command $\vec{a}$ during a time interval $[u_0, u_t]$, then the formula ϕ_{E_M} includes universal quantification over time along with solutions of the ODEs, such as a formula of the form:

$$\bigvee \left(guard(\vec{a}) \to \forall t \in [u_0, u_t].\ \vec{x}(t) = \vec{v} + \int_0^{t-u_0} F_{\vec{a}}(\vec{x}, t)\,\mathrm{d}t \right)$$

Definition 7. *For $E_M = (C, \vec{x}, \Lambda)$, $\phi_{E_M}(\vec{a}, u_0, u_t, \vec{v} \shortmid \vec{x})$ iff $((\vec{a}, u_t - u_0, \vec{v}), \vec{\tau}) \in \Lambda$ and $\vec{x}(t) = \vec{\tau}(t - u_0)$ for $t \in [u_0, u_t]$.*

The encoding $\phi_{M\restriction_\pi E_M}$ of the environment restriction $M \restriction_\pi E_M$ has the form $\phi^{T,i}_{M\restriction_\pi E_M}(\vec{i}, \vec{y}, \vec{v} \shortmid \vec{y'}, \vec{v'}, \vec{o})$, with variables: (i) $\vec{i}$ denoting input, (ii) $(\vec{y}, \vec{v})$ denoting a state at the beginning of the round (at time $iT - \epsilon$), (iii) $(\vec{y'}, \vec{v'})$ denoting a state at the end of the round (at time $(i+1)T - \epsilon$), and (iv) $\vec{o}$ denoting output. Instead of writing $iT - \epsilon$ and $(i+1)T - \epsilon$ in $\phi_{M\restriction_\pi E_M}$, we use the "$\epsilon$-shifted" time axis for the interval $[iT, (i+1)T]$. Due to clock skews, the exact values of sampling time $u_I = (c_M(i) + t_I) - (iT - \epsilon)$ and response time $u_R = (c_M(i) + t_R) - (iT - \epsilon)$ are unknown. Because $iT - \epsilon < c_M(i) < iT + \epsilon$, we represent those times as formulas $t_I < u_I < t_I + 2\epsilon$ and $t_R < u_R < t_R + 2\epsilon$ in $\phi_{M\restriction_\pi E_M}$.

Definition 8. *For an environment restriction $M \restriction_\pi E_M$ with interface π, $\phi^{T,i}_{M\restriction_\pi E_M}(\vec{i}, \vec{y}, \vec{v} \shortmid \vec{y'}, \vec{v'}, \vec{o})$ is defined by:*

$$\begin{aligned}
&\exists \vec{a}, \vec{a'}, \vec{v}_I, \vec{v}_R, u_I, u_R, t_I, t_R. && \vec{a} = \pi_C(\vec{y}) \\
&\wedge\ \phi_{E_M}(\vec{a}, iT, iT + u_I, \vec{v} \shortmid \vec{x}) && \wedge\ \vec{v}_I = \vec{x}(iT + u_I) \\
&\wedge\ \phi_{E_M}(\vec{a}, iT + u_I, iT + u_R, \vec{v}_I \shortmid \vec{x}) && \wedge\ \vec{v}_R = \vec{x}(iT + u_R) \\
&\wedge\ \phi_M(\vec{i}, \langle \vec{y}, \pi_O(\vec{v}_I) \rangle \shortmid \langle \vec{y'}, \pi_O(\vec{v'}) \rangle, \vec{o}) && \wedge\ \vec{a'} = \pi_C(\vec{y'}) \\
&\wedge\ \phi_{E_M}(\vec{a'}, iT + u_R, (i+1)T, \vec{v}_R \shortmid \vec{x}) && \wedge\ \vec{v'} = \vec{x}((i+1)T) \\
&\wedge\ t_I = \pi_I(\vec{y}) && \wedge\ t_R = \pi_R(\vec{y}, \vec{i}) \\
&\wedge\ (t_I < u_I < t_I + 2\epsilon) && \wedge\ (t_R < u_R < t_R + 2\epsilon)
\end{aligned}$$

Notice that a local clock c_M (and the round number π_T) are abstracted from the formula. That is, $\phi_{M\restriction_\pi E_M}(\vec{i}, s, \vec{v} \shortmid s', \vec{v'}, \vec{o})$ iff $((\vec{i}, (s, \vec{v})), ((s', \vec{v'}), \vec{o})) \in \delta_{M\restriction_\pi E_M}$ for some c_M.

B. Encoding Hybrid Ensembles

The synchronous composition of an ensemble $\mathcal{E}$ is equivalent to a single machine $M_\mathcal{E}$ in which all machines perform their transitions in lockstep. Each fast machine f performs $k = rate(f)$ internal transitions in one step. The encoding $\phi^{T,i}_{(M\restriction_\pi E_M)^{\times k}}$ of the k-step *deceleration* of $M \restriction_\pi E_M$ is defined by sequentially composing the k formulas $\phi^{T/k,ik}_{M\restriction_\pi E_M}, \ldots, \phi^{T/k,(ik+k-1)}_{M\restriction_\pi E_M}$ corresponding to the k *subintervals* $[(ik + n - 1)T/k - \epsilon, (ik + n)T/k - \epsilon]$ for $n = 1, \ldots, k$.

Definition 9. *Given $M \restriction_\pi E_M$ and $k \in \mathbb{N}$, the formula $\phi^{T,i}_{(M\restriction_\pi E_M)^{\times k}}(\langle \vec{i_1}, \ldots, \vec{i_k} \rangle, \vec{y_0}, \vec{v_0} \shortmid \vec{y_k}, \vec{v_k}, \langle \vec{o_1}, \ldots, \vec{o_k} \rangle)$ is:*

$$\exists \{\vec{y_n}, \vec{v_n}\}_{n=1}^{k-1}.\ \bigwedge_{n=1}^{k} \phi^{T/k,(ik+n-1)}_{M\restriction_\pi E_M}(\vec{i_n}, \vec{y}_{n-1}, \vec{v}_{n-1} \shortmid \vec{y_n}, \vec{v_n}, \vec{o}_n)$$

The wiring diagram of $\mathcal{E}$ is encoded as a conjunction of equalities between variables denoting input and output ports. Each equality corresponds to a connection in $\mathcal{E}$ (together with an input adaptor for machines with different rates).

Definition 10. *ϕ_{wire} is the conjunction of the equalities: (i) $\alpha^l_j(i^l_j) = i^n_e$ for each connection from $\mathcal{E}$'s n-th input port to M_j's l-th input port with its input adaptor α^l_j; (ii) $o^n_e = o^l_j$ for each connection from M_j's l-th output port to $\mathcal{E}$'s n-th output port; and (iii) $\alpha^l_j(i^l_j) = f^n_k$ and $f^{n\prime}_k = o^n_k$ for each connection from M_k's n-th output port to M_j's l-th input port with its input adaptor α^l_j, where f^l_k denotes a feedback output from the previous step, and $f^{l\prime}_k$ denotes one for the next step.*

A hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ of machines $\{M_j\}_{j \in J_S \cup J_F}$ with environments $\{E_{M_j}\}_{j \in J_S \cup J_F}$ (J_S denoting slow machines and J_F denoting fast machines) is encoded as a formula $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\vec{i}, \{\vec{z_j}, \vec{f_j}\}_{j \in J_S \cup J_F} \shortmid \{\vec{z'_j}, \vec{f'_j}\}_{j \in J_S \cup J_F}, \vec{o})$, with variables $\vec{i}$, $\vec{z_j}$, $\vec{f_j}$, $\vec{z'_j}$, $\vec{f'_j}$, and $\vec{o}$ denoting, respectively, inputs, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the beginning of the round (at time $iT - \epsilon$), feedback outputs from the previous round, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the end of the round (at time $(i+1)T - \epsilon$), the feedback outputs for the next round, and the output.

Definition 11. *For a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$, its encoding $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\vec{i}, \{\vec{y_j}, \vec{v_j}, \vec{f_j}\}_{j\in J_S\cup J_F} \restriction \{\vec{y'_j}, \vec{v'_j}, \vec{f'_j}\}_{j\in J_S\cup J_F}, \vec{o})$ is:*

$$\exists\{\vec{i_j}, \vec{o_j}\}_{j\in J_S\cup J_F}.\ \textstyle\bigwedge_{s\in J_S}\big(\phi^{T,i}_{M_s\restriction_{\pi_s}E_{M_s}}(\vec{i_s}, \vec{y_s}, \vec{v_s} \restriction \vec{y'_s}, \vec{v'_s}, \vec{o_s})\big)$$
$$\wedge\ \textstyle\bigwedge_{f\in J_F}\big(\phi^{T,i}_{(M_f\restriction_{\pi_f}E_{M_f})^{\times rate(j)}}(\vec{i_f}, \vec{y_f}, \vec{v_f} \restriction \vec{y'_f}, \vec{v'_f}, \vec{o_f})\big)$$
$$\wedge\ \phi_{wire}(\vec{i}, \vec{o}, \{\vec{i_j}, \vec{o_j}, \vec{f_j}, \vec{f'_j}\}_{j\in J_S\cup J_F})\ \wedge\ (\forall t.\ \psi).$$

By construction, a formula $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable iff there is a corresponding transition of the synchronous composition $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ from $iT-\epsilon$ to $(i+1)T-\epsilon$ for *some* local clocks. Therefore, by the bisimulation equivalence (Theorem 2):

Theorem 3. *$\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable iff for some local clocks, there exists a corresponding* stable transition *in a distributed hybrid system $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ for $[iT-\epsilon, (i+1)T-\epsilon]$.*

C. Encoding Verification Problems

Our goal is to verify safety properties of a distributed CPS $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$. In general, a safety property of the system can be expressed as a formula of the form $\forall t.safe(\vec{z}, t)$ for state variables $\vec{z}$ and time variable t.

To verify a safety property up to a given bound $n \in \mathbb{N}$ (i.e., for the time interval $[0, nT-\epsilon]$), we encode its *bounded counterexamples*. If the formula is unsatisfiable (i.e., no counterexample exists), then, by Theorem 3, the system satisfies the safety property in $[0, nT-\epsilon]$ *for any local clocks*.

Definition 12. *The verification problem for $\forall t.safe(\vec{z}, t)$* up to n rounds *with initial condition $init(\vec{z})$ is encoded by:*

$$\exists\vec{z}_0, \{\vec{z}_k, \vec{i}_k, \vec{o}_k, t_k\}_{k=1}^n.\ \textstyle\bigwedge_{k=1}^n\big(\phi^{T,k}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\vec{i}_k, \vec{z}_{k-1} \restriction \vec{z}_k, \vec{o}_k)\big)$$
$$\wedge\ init(\vec{z}_0)\ \wedge\ \textstyle\bigvee_{k=1}^n\big((k-1)T < t_k \le T \wedge \neg safe(\vec{z}_k, t_k)\big).$$

For *unbounded* verification of a safety property, we encode its inductive proof as a logical formula. The idea is to find an inductive condition $ind(\vec{z})$ for synchronous transitions that corresponds to the beginning of a global round, and to show that $ind(\vec{z})$ implies $safe(\vec{z}, t)$ for that round.

Definition 13. *The verification problem for $\forall t.safe(\vec{z}, t)$ with an inductive condition $ind(\vec{z})$ is encoded by:*

$$\forall\vec{z}, \vec{z'}, \vec{i}, \vec{o}.\ (ind(\vec{z})\ \wedge\ \phi^{T,0}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\vec{i}, \vec{z} \restriction \vec{z'}, \vec{o}))$$
$$\implies ind(\vec{z'})\ \wedge\ t \in [0, T].\ safe(\vec{z}, t).$$

We also encode a divide-and-conquer proof to verify a safety property by compositional analysis. An input condition $c_{in}(\vec{i}, \vec{z}, t)$ and an output condition $c_{out}(\vec{o}, \vec{z}, t)$ during a global round are identified for synchronous transitions in such a way that $c_{in}(\vec{i}, \vec{z}, t)$ implies $safe(\vec{z}, t)$ for that round.

Definition 14. *The compositional analysis for $\forall t.safe(\vec{z}, t)$ with I/O conditions $c_{in}(\vec{i}, \vec{z}, t)$ and $c_{out}(\vec{o}, \vec{z}, t)$ is encoded by:*

$$\forall\vec{z}, \vec{z'}, \vec{i}, \vec{o}.\ (c_{in}(\vec{i}, \vec{z}, t)\ \wedge\ \phi^{T,0}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\vec{i}, \vec{z} \restriction \vec{z'}, \vec{o}))$$
$$\implies c_{out}(\vec{o}, \vec{z'}, t)\ \wedge\ t \in [0, T].\ safe(\vec{z}, t).$$

The validity of such formulas for inductive or compositional analysis can also be automatically checked using SMT solving by checking the unsatisfiability of their *negations*.

V. SMT Solving for Distributed CPSs

Section IV shows that the verification of distributed CPSs can be encoded as logical formulas over the real numbers and ODEs. It would therefore be desirable to use SMT solving to automatically check the satisfiability of those formulas. The satisfiability of such formulas is undecidable for nonlinear hybrid systems, but becomes *decidable* up to any given precision $\delta > 0$. A *δ-complete decision procedure* for a formula ϕ returns false if ϕ is unsatisfiable, and returns true if its *syntactic numerical perturbation* of ϕ by bound δ is satisfiable.[2] This is very useful in practice, since sampling exact values of physical parameters is not possible in reality.

However, formulas encoding Hybrid PALS models may contain ODEs and universal quantification over *uninterpreted functions on reals*, such as $\forall t \in [u_0, u_t].\ \vec{x}(t) = \vec{v} + \int_0^{t-u_0} F_{\vec{a}}(\vec{x}, t)\,\mathrm{d}t$ or time-invariant constraints $(\forall t.\ \psi)$, which are not supported by current state-of-the-art SMT solving techniques. This section therefore shows how formulas for hybrid PALS models (presented in Section IV) can be equivalently encoded as *SMT formulas without* universal quantification over uninterpreted real functions. For this purpose, we present a new SMT framework to provide an efficient SMT algorithm for formulas encoding Hybrid PALS verification problems.

A. Theory of the Real Numbers with Function Names

SMT-based techniques for hybrid systems normally use the standard SMT theory of the real numbers and computable real functions, such as polynomials, exponentiation, trigonometric functions, and solutions of Lipschitz-continuous ODEs.

Definition 15. *For a finite set $\mathcal{F}$ of computable real functions (with real number constants seen as 0-ary functions), $\mathcal{L}_\mathcal{F} = (\mathcal{F}, >)$ denotes the first-order signature over the real numbers with the functions in $\mathcal{F}$, and $\mathbb{R}_\mathcal{F} = (\mathbb{R}, \mathcal{F}^\mathbb{R}, >^\mathbb{R})$ is the standard structure of the theory of the real numbers.*

Solutions of ODEs are considered *atomic functions* in $\mathcal{L}_\mathcal{F}$, and therefore the structure of ODEs cannot be used for SMT algorithms in this logic. This restrictive syntax makes it difficult to efficiently encode logical formulas for Hybrid PALS in $\mathcal{L}_\mathcal{F}$. We therefore present a new SMT theory, by extending $\mathcal{L}_\mathcal{F}$, that allows to express solutions of ODEs in a modular way, so that the size of the formula can be significantly reduced.

We consider a *two-sorted* first-order logic with sorts $Real$ and $Name$, where $Real$ denotes the real numbers, and $Name$ denotes *name constants* for unary functions composed of the functions in $\mathcal{F}$. For example, given $\{1, +, \times, \sin, x, y\} \subset \mathcal{F}$ and two name constants $\mathtt{m}_1$ and $\mathtt{m}_2$ of sort $Name$, we can have two *named* real functions: $[\mathtt{m}_1] : \sin(x(t)) : Real \to Real$ and $[\mathtt{m}_2] : [y(t)+1, z(t)^2] : Real \to Real^2$.

The new logic includes a collection of *application operators* $app^n : Name \times Real \to Real^n$ that connect a name constant to its underlying function with range $Real^n$, e.g.,

$$app^1(\mathtt{m}_1, 3) \equiv \sin(x(3)), \quad app^2(\mathtt{m}_2, u) \equiv [y(u)+1, z(u)^2].$$

[2] E.g., if $\psi \equiv (x > 3) \wedge (y = z)$, then its syntactic numerical perturbation by $\delta > 0$ is $(x - 3 > -\delta) \wedge (y - z \ge -\delta) \wedge (z - y \ge -\delta)$.

In the new logic, we take into account a collection of *integral operators* $int^{k_1,\ldots,k_n} : Real \times Name^n \to Real^{\sum_{i=1}^n k_i}$. An integral term $int^{k_1,\ldots,k_n}(u, \nu_1, \ldots, \nu_n)$ takes time value u and a list of name constants $\nu_1, \ldots, \nu_n$, to respectively denote real functions $f_1, \ldots, f_n$ with ranges $Real^{k_1}, \ldots, Real^{k_n}$, and returns the value $\int_0^u [f_1(t), \ldots, f_n(t)]\,\mathrm{d}t$. For example:

$$int^2(1, \mathtt{m_2}) \equiv \int_0^1 [y+1, z^2]\mathrm{d}t,$$
$$int^{1,2}(T, \mathtt{m_1}, \mathtt{m_2}) \equiv \int_0^T [\sin(x), y+1, z^2]\mathrm{d}t.$$

Definition 16. *For a set $\mathcal{N}$ of name constants and a set $\mathcal{O}$ of operators app^n and $int^{k_1,\ldots,k_n}$, the first order signature is given by $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}} = (\mathcal{F}\cup\mathcal{N}\cup\mathcal{O}, >)$. The first order structure is given by $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}} = (\mathbb{R}\cup N, \mathcal{F}^{\mathbb{R}}\cup\mathcal{N}^N\cup\mathcal{O}^{N,\mathbb{R}}, >^{\mathbb{R}})$ for a fixed finite set N of* name objects*, where the interpretation $\mathcal{N}^N$ of name constants and the interpretation $\mathcal{O}^{N,\mathbb{R}}$ of application and integral operators are given as explained above. The syntax and semantics of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas is defined by means of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ and $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ in the standard way.*

The satisfiability problems of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formulas in the new theory $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ can be reduced to ones in the standard theory $\mathbb{R}_{\mathcal{F}}$ at minimal cost, provided that all $Name$ variables are only existentially quantified. Consider an $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ formula $\exists\vec{n}.\ \psi(\vec{n})$, where $\vec{n}$ are only variables of sort $Name$ and $\psi(\vec{n})$ contains no quantifier for $\vec{n}$. Since there are only a finite number of name objects in $\mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$, by using the standard SMT solving for equalities, we can enumerate all *consistent* assignments $\vec{n} = \overrightarrow{name}_1, \ldots, \vec{n} = \overrightarrow{name}_N$ for $\vec{n}$. Let $\hat{\psi}_i$ be the formula obtained from $\psi(\vec{n})$ by replacing each name by its related function according to the i-th assignment $\vec{n} = \overrightarrow{name}_i$. Then, $\vee_{i=1}^N \hat{\psi}_i$ is an ordinary $\mathcal{L}_{\mathcal{F}}$ formula whose satisfiability can be decided by δ-complete decision procedures. Therefore:

Theorem 4. *The satisfiability of $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ formulas of the form $\exists\vec{n}.\ \psi(\vec{n})$ is decidable by δ-complete SMT solving.*

B. SMT Encoding of Hybrid PALS Models

We now show how Hybrid PALS models can be encoded in the new SMT logic $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ *without* using uninterpreted real functions and universal quantification. The idea is to construct a *global* ODE formula ϕ_{ODE} that defines the combined systems of ODEs for all physical environments in $\mathcal{E} \restriction_\Pi E_\mathcal{E}$. The formula ϕ_{ODE} is parameterized by the behaviors of the subcomponents by means of integral operators and *name variables*. The behavior of each subcomponent $M_j \restriction_{\pi_j} E_{M_j}$ is then specified in a modular way by assigning to each name variable an appropriate name constant for $M_j \restriction_{\pi_j} E_{M_j}$.

We restrict our attention to time-invariant constraints with only equality terms, since physical correlations for CPS in practice typically can be expressed using equalities. Equality constraints, such as $x_1(t) = x_2(t)$, can be removed from the formula by replacing one side with the other, for example, by replacing each function symbol x_1 with x_2. From now on we assume that time-invariant equality constraints have been removed from the formula in this way.

Suppose that $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ includes N terms of the form $\vec{x_j}(u)$ to obtain a value of some parameter of E_{M_j}. A global ODE formula ϕ_{ODE} is defined as a conjunction of N ODEs for these access times that include all sampling and response times for each (decelerated) subcomponent in a global round.

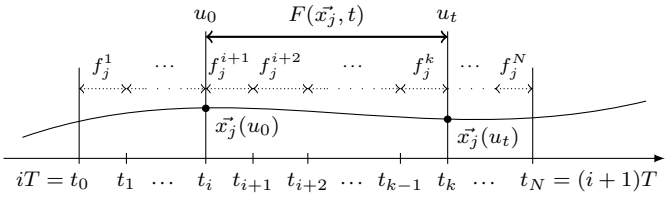

Fig. 6. $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-transformation for ODEs

Definition 17. *Let (i) $\vec{t} = \{t_0, t_1, \ldots, t_N\}$ be N* time variables *with $iT = t_0 \le t_1 \le t_2 \le \cdots \le t_N = (i+1)T$; (ii) $\vec{f} = \{f_j^1, \ldots, f_j^N\}_{j\in J_S\cup J_F}$ be* name variables *denoting E_{M_j}'s continuous behaviors during each interval $[t_{i-1}, t_i]$; (iii) $\vec{g} = \{\vec{g}_0, \ldots, \vec{g}_N\}$ be* global value variables *denoting the values of all physical parameters of $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ at each time t_i; and (iv) $\vec{l} = \{l_j\}_{j\in J_S\cup J_F}$ denote the* dimension *of each E_{M_j}'s parameters $\vec{x_j} = (x_j^1, \ldots, x_j^{l_j})$. The formula $\phi_{ODE}(\vec{f}, \vec{t}, \vec{g})$ is: $\bigwedge_{1\le i\le N} (\vec{g}_i = \vec{g}_{i-1} + int^{\vec{l}}(t_i - t_{i-1}, \{f_j^i\}_{j\in J_S\cup J_F}))$.*

For any subcomponent $M_j \restriction_{\pi_j} E_{M_j}$, an ODE subformula $\forall t \in [u_0, u_t].\ \vec{x_j}(t) = \vec{v_j} + \int_0^{t-u_0} F(\vec{x_j}, t)\,\mathrm{d}t$ is replaced by conditions for variables $\vec{f}$, $\vec{t}$, and $\vec{g}$. For example, if $u_0 = t_i$ and $u_t = t_k$ for some $0 \le i \le k \le n$, then all f_j^l for $i < l \le k$ denote the function $F(\vec{x_j}, t)$, as illustrated in Fig. 6.

Definition 18. *Let $\pi_j(\vec{g}_i)$ denote E_{M_j}'s parameter values in the global values $\vec{g}_i$. Consider an ODE formula of the form $\forall t \in [u_0, u_t].\ \vec{x_j}(t) = \vec{v_j} + \int_0^{t-u_0} F(\vec{x_j}, t)\,\mathrm{d}t$. If a name constant $\mathtt{m}_{F(\vec{x_j},t)}$ denotes $F(\vec{x_j}, t)$, its $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formula is:*

$$\bigvee_{0\le i\le k\le n} \begin{bmatrix} (t_i = u_0) \wedge (t_k = u_t) \wedge \bigwedge_{l=i+1}^k f_j^l \equiv \mathtt{m}_{F(\vec{x_j},t)}) \\ \wedge\ (\vec{x_j}(u_0) = \pi_j(\vec{g}_i) = \vec{v_j}) \wedge (\vec{x_j}(u_t) = \pi_j(\vec{g}_k)) \end{bmatrix}$$

Let $tr(\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}})$ be the formula obtained from $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ by repeatedly performing the $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-transformation for every ODE subformula. Finally, we obtain the $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-formula $\tilde{\phi}^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}} \equiv \exists(\vec{f}, \vec{t}, \vec{g}).\ \phi_{ODE}(\vec{f}, \vec{t}, \vec{g}) \wedge tr(\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}})$ that includes no uninterpreted real functions and universal quantification. Therefore, the satisfiability of $\tilde{\phi}^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ can be determined by using δ-complete SMT solving (by Theorem 4).

Theorem 5. *$\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable iff $\tilde{\phi}^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable.*

Proof Sketch. ($\Rightarrow$) If $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable, then there exists a collection $\{\vec{x_j}\}_{j\in J_S\cup J_F}$ of trajectories for a global round, and we can find the values of $(\vec{f}, \vec{t}, \vec{g})$ from the trajectories. ($\Leftarrow$) If $\tilde{\phi}^{T,i}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable, then there exist resulting values of $(\vec{f}, \vec{t}, \vec{g})$. Then, by construction, any ODE subformula $\forall t \in [u_0, u_t].\ \vec{x_j}(t) = \vec{v_j} + \int_0^{t-u_0} F(\vec{x_j}, t)\,\mathrm{d}t$ is true iff its transformed formula by Definition 18 is true, provided that $u_0 = t_i$ and $u_t = t_k$ for some $0 \le i \le k \le n$. □

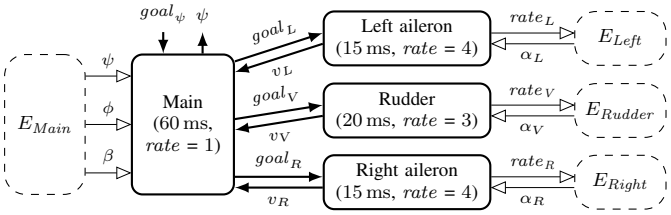

Fig. 7. The distributed controllers for turning an airplane.

VI. Case Studies and Experimental Results

This section gives an overview of some case studies using Hybrid PALS and SMT-based analysis to verify distributed CPSs. These case studies involve nontrivial (nonlinear) ODEs, due to the continuous physical interaction between distributed components. We have verified safety properties using inductive and compositional SMT encodings for *any possible* set of local clocks maximal clock skew ϵ. All experiments in this section were conducted on an Intel Xeon 2.0 GHz with 64 GB memory. The case studies and the experimental evaluation are available at http://dreal.github.io/benchmarks/networks.

A. Turning an Airplane

We consider a multirate distributed CPS to turn an airplane (adapted from [9]). To make a turn, an aircraft rolls towards the direction of the turn by moving its *ailerons* The rolling causes a yawing moment in the opposite direction, called *adverse yaw*, which is countered by using its *rudder* (a surface attached to the vertical stabilizer). The subcontrollers for the ailerons and the rudder operate at different rates, and the main controller orchestrates them to achieve a smooth turn. Fig. 7 illustrates the multirate ensemble $\mathcal{E}$ of our system.

Each subcontroller M gradually moves its surface towards the goal angle $goal_M$ specified by the main controller M_{Main}. In each round, M receives $goal_M$ from M_{Main}, determines the moving rate $rate_M$ based on $goal_M$ and the current sampled value v_M of the surface angle α_M, and sends back v_M to M_{Main} (e.g., the new value of $rate_M$ can be given by $\text{sign}(goal_M - v_M) \cdot \min(\text{abs}(goal_M - v_M)/T, max_M)$). The continuous dynamics of α_M is specified by the local physical environment E_M as the ODE $\frac{\mathrm{d}\alpha_M}{\mathrm{d}t} = rate_M$.

The main controller M_{Main} determines the goal angles for the subcontrollers to make a coordinated turn. In each round, M_{Main} receives a desired direction $goal_\psi$ (from the pilot) and the surface angles (v_L, v_V, v_R) from the subcontrollers, and then sends back the new goal angles $(goal_L, goal_V, goal_R)$, based on the current sampled *position* values $(v_\psi, v_\phi, v_\beta)$ of the direction angle ψ, the roll angle ϕ, and the yaw angle β.

We consider a simple control logic to decide the new goal angles $(goal_L, goal_V, goal_R)$, by three control functions [9]: (i) $f_\phi(v_\phi, v_\psi, goal_\psi)$ returns the goal roll angle $goal_\phi$, (ii) $f_R(v_\phi, goal_\phi)$ returns the goal angle $goal_R$ for the right aileron (where the goal angle $goal_L$ for the left aileron is the opposite angle $-goal_R$); and (iii) $f_V(v_\beta)$ returns the goal rudder angle $goal_V$. For example, for $h = 0.32(g_\psi - v_\psi)$:

$$
\begin{aligned}
f_\phi(v_\phi, v_\psi, g_\psi) &= v_\phi + \text{sign}(h - v_\phi) \cdot \min(\text{abs}(h - v_\phi), 1.5) \\
f_R(v_\phi, g_\phi) &= \text{sign}(g_\phi - v_\phi) \cdot \min(0.3\,\text{abs}(g_\phi - v_\phi), 45)) \\
f_V(v_\beta) &= \text{sign}(-v_\beta) \cdot \min(0.3\,\text{abs}(v_\beta), 30).
\end{aligned}
$$

In the physical environment E_{Main}, the lateral dynamics of an aircraft can be specified as the following nonlinear ODEs, where p is the rolling moment, r is the yawing moment, and $Y_{\delta_L,\delta_V,\delta_R,\beta}$, $L_{\delta_L,\delta_V,\delta_R,\beta}$, and $N_{\delta_L,\delta_V,\delta_R,\beta}$ are (linear) functions of the control angles $(\delta_L, \delta_V, \delta_R)$ and β [10]:

$$
\begin{aligned}
&\dot\beta = Y_{\delta_L,\delta_V,\delta_R,\beta}/mV - r + (g/V)\cos\beta\sin\phi, \\
&\dot\phi = p, \qquad\qquad \dot\psi = (g/V)\tan\phi, \\
&\dot p = (c_1 r + c_2 p)\cdot r\tan\phi + c_3 L_{\delta_L,\delta_V,\delta_R,\beta} + c_4 N_{\delta_L,\delta_V,\delta_R,\beta}, \\
&\dot r = (c_8 p - c_2 r)\cdot r\tan\phi + c_4 L_{\delta_L,\delta_V,\delta_R,\beta} + c_9 N_{\delta_L,\delta_V,\delta_R,\beta}.
\end{aligned}
$$

The physical environment E_{Main} clearly depends on the subcontrollers's physical environments. Each control angle δ_M in E_{Main} must be the same as the corresponding surface angle α_M. Such immediate physical correlations between the local environments are specified by the time-invariant constraint

$$\forall t.\, (\delta_L(t) = \alpha_L(t)) \wedge (\delta_V(t) = \alpha_V(t)) \wedge (\delta_R(t) = \alpha_R(t)).$$

Notice that since the main controller and the subcontrollers have *different periods with local clock skews*, the ODEs of the subcontrollers cannot be directly "plugged" into E_{Main}.

The safety property of $\mathcal{E}$ is that the yaw angle β is always close to 0 (e.g., $\forall t.\ \beta(t) < 0.1$). In the analysis, we assume that the maximal clock skew is $\epsilon = 0.1\,\text{ms}$, the sampling time t_I is 0 ms for every controller, and the response time t_R is 3 ms for the subcontrollers. **Working on the verification.**

B. Networked Water Tank Controllers

In this case study, adapted from [11], [12], a number of water tanks are connected by pipes as shown in Figure 8. **KYUNGMIN: I cannot see on Fig 8 that they are connected ...** The water level in each tank is controlled by a pump in the tank, and depends on the pump's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the water levels of the adjacent tanks. The water level x_i of tank i changes according to the ODEs:[3]

$$
\begin{aligned}
A_i\dot x_i &= (q_i + a\sqrt{2g}\sqrt{x_{i-1}}) - b\sqrt{2g}\sqrt{x_i} \quad \text{if } m_i = m_{\texttt{on}}, \\
A_i\dot x_i &= a\sqrt{2g}\sqrt{x_{i-1}} - b\sqrt{2g}\sqrt{x_i} \quad \text{if } m_i = m_{\texttt{off}},
\end{aligned}
$$

We set $x_0 = 0$ for the leftmost tank 1. Every pipe controller performs its transitions according to its local clock and sets the pump to on if $x_i \le L_m$ and to off if $x_i > L_M$.

The safety property is that the water level of each tank is in a certain range $I = [L_m - \eta, L_M + \eta]$. We have verified this safety property for *any number* of connected water tanks

[3] A_i, q_i, a, b are constants determined by the size of the tank, the power of the pump, and the widths of the I/O pipes, and g is the gravity constant.

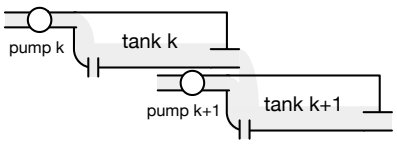

Fig. 8. Connected water tanks.

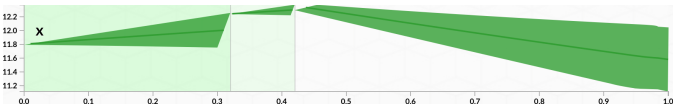

Fig. 9. The counterexample trajectory when $\epsilon = 150\,\text{ms}$, where the water level increases for extra 300 ms to violate the inductive condition.

using inductive and compositional analysis.[4] For maximal clock skew $\epsilon = 30\,\text{ms}$, sampling time $t_I = 20\,\text{ms}$, and response time $t_R = 100\,\text{ms}$, we have proved the compositional safety property using dReal with precision $\delta = 0.001$ (the analysis took 4.3 seconds).[5] However, if $\epsilon = 150\,\text{ms}$, then the compositional inductive condition is violated by the trajectory in Fig. 9 (the analysis took 1.46 seconds).

C. Networked Thermostat Controllers

A number of rooms are interconnected by open doors, as shown in Fig. 10. The temperature of each room is separately controlled by its own thermostat controller that turns the heater on and off. That is, it depends on the heater's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the temperatures of the other rooms. The temperature x_i of room i changes according to the ODEs:[6]

$$\dot{x}_i = K_i(h_i - ((1-2c)x_i + cx_{i-1} + cx_{i+1})) \quad \text{if } m_i = m_{\texttt{on}}$$
$$\dot{x}_i = -K_i((1-2c)x_i + cx_{i-1} + cx_{i+1}) \quad \text{if } m_i = m_{\texttt{off}}$$

In each transition, a controller turns on the heater if $x_i \le T_m$, and turns it off if $x_i > T_M$.

The safety property is that the temperature of each room is in the range $I = [T_m - \eta, T_M + \eta]$. We have verified the safety property $\forall t.\ x_i \in I$ for *any number* of interconnected thermostat controllers by inductive and compositional analysis.[7] For

[4]For a tank k and $I' = [L_m - \eta', L_M + \eta'] \subseteq I$ with $\eta' < \eta$, provided that $x_{k-1} \in I$ always holds, we show that $x_k \in I'$ is an inductive condition, and $x_k \in I$ always holds if $x_k \in I'$ at the beginning of each round (i.e., $(x_k(0) \in I' \wedge \phi^{T,0}_{\mathcal{E}\restriction_{\Pi E_{\mathcal{E}}}}) \to (x_k(T) \in I') \wedge (\forall t \in [0,T].\ x_k(t) \in I)$).

[5]In the analysis, we use the parameters $a = 0.5$, $b = 0.6$, $g = 9.8$, $A = 2$, $q = 4$, $L_m = 8$, $L_M = 10$, $\eta = 3$, $\eta' = 2$, and $T = 1\,\text{s}$.

[6]$K_i, h_i \in \mathbb{R}$ are constants depending on the size of room i and the heater's power, respectively, and $c \in \mathbb{R}$ is determined by the size of the open door.

[7]For $I' = [T_m - \eta', T_M + \eta'] \subseteq I$, provided that both $x_{i-1} \in I$ and $x_{i+1} \in I$ always hold, we show that $x_i \in I'$ is an inductive condition, and $x_i \in I$ always holds if $x_i \in I'$ at the beginning of each round.

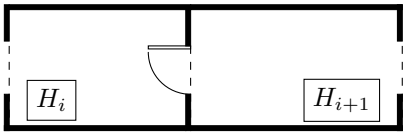

Fig. 10. Interconnected rooms.

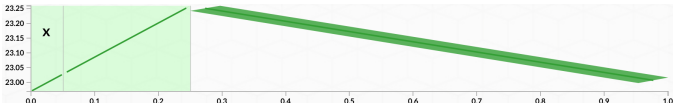

Fig. 11. The counterexample trajectory when $\epsilon = 20\,\text{ms}$.

$\epsilon = 2\,\text{ms}$, $t_I = 10\,\text{ms}$, and $t_R = 200\,\text{ms}$, we have proved (in 2.6 seconds) the compositional safety property using dReal with precision $\delta = 0.001$.[8] However, if $\epsilon = 20\,\text{ms}$, then the compositional inductive condition is violated by the trajectory in Fig. 11 (the analysis took 0.56 seconds).

D. Bounded Reachability Comparison

We have implemented the SMT algorithm for our new logic $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ in the dReal SMT solver [8], and have compared the performance of the new $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$-encoding with one of the previous *non-modular* $\mathcal{L}_{\mathcal{F}}$-encoding for Hybrid PALS models. In this comparison, we only consider special cases with no clock skews ($\epsilon = 0$) due to a lack of expressiveness of $\mathcal{L}_{\mathcal{F}}$, and only bounded reachability analysis since it needs to generate bigger formulas than other types of analysis.

We have performed bounded reachability analysis up to $k = 5$ for the safety properties for the "concrete" instances of the case studies (using dReal without the use of other SMT heuristics). The results for k-step bounded reachability analysis are summarized in Fig. 12 and show that the new encoding significantly outperforms the non-modular encoding.[9] For example, the SMT analysis using the non-modular encoding for three interconnected thermostats did not terminate for 30 hours even for $k = 2$.

VII. Related Work

PALS [1]–[4] targets distributed real-time systems, whose absence of continuous behaviors means that continuous behaviors and local clocks do not need to be taken into account in the synchronous models, which can therefore be verified by any explicit-state model checker. In contrast, Hybrid PALS synchronous models must take both clock skews and continuous behaviors into account and hence cannot be analyzed by explicit-state techniques. The first steps to add continuous behaviors to PALS were taken in [5]. Our work presents a more general model, where the relative sampling and actuating times are system parameters, and also provides a crucial bisimulation result between the synchronous and the distributed model.

However, the main difference is in the verification part. Previous PALS work [9] used explicit-state model checking, based on numerical simulation for the continuous dynamics. Obviously, this cannot guarantee the correctness of hybrid systems. Furthermore, there was no hope of verifying a system for all possible local clocks. The paper [5] shows that two interconnected thermostats can be verified using dReal, but does

[8]In the analysis, we use the parameters $K = 0.015$, $h = 100$, $c = 0.01$, $T_M = 21$, $T_m = 19$, $\eta = 3$, $\eta' = 2$, and $T = 1\,\text{s}$.

[9]We consider both `unsat` (verified) and `sat` (counterexample found) cases by using different safety properties.

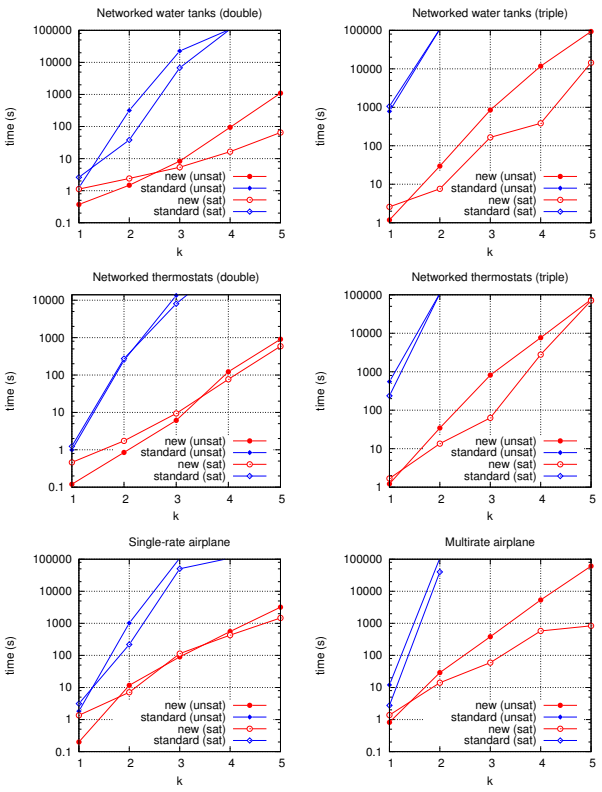

Fig. 12. Running time of k-step bounded reachability analysis, where red lines denote the new encoding, and blue lines denote the non-modular encoding.

not present any general SMT techniques for Hybrid PALS. Our work presents SMT encodings for bounded reachability, inductive, and compositional analysis, besides a new SMT framework and more case studies.

Because of the difficulty of handling SMT formulas over the reals with nonlinear functions, SMT-solving-based verification is a fairly new direction for nonlinear hybrid systems. The research direction is initiated in [13], which uses constraint solving algorithms for handling nonlinear reachability problems. Two main lines of work that explicitly formulate problems as SMT formulas are based on the HySAT/iSAT solver [14], [15] and the MathSAT solver [16], [17]. But neither efficient encodings of distributed hybrid systems nor inductive and compositional analysis has been investigated in existing work along these lines. It is also worth noting that our methods are orthogonal to reachable set computation tools like SpaceEX [18] and Flow* [19], since they can also be used as ODE solvers for δ-compete SMT (e.g., in dReal).

VIII. Concluding Remarks

We have presented and proved the correctness of the Hybrid PALS methodology that greatly simplifies the design and verification of virtually synchronous cyber-physical systems whose components have environments with continuous behaviors. Although Hybrid PALS does not allow us to abstract from (imprecise) local clocks and the times transitions are performed, it allows us to abstract from asynchronous communication (and the resulting interleavings), message buffering, network delays, backoff timers, and so on.

We have shown that verification problems for Hybrid PALS models can be expressed as SMT formulas and have developed efficient SMT-solving-based verification methods for Hybrid PALS. We have implemented these techniques in the dReal SMT solver and have applied our methodology on a number of non-trivial CPSs. Our experiments have shown that our techniques dramatically increase the performance of SMT analysis for distributed hybrid systems with multiple control modes and nonlinear ODEs up to precision δ.

References

[1] A. Al-Nayeem, M. Sun, X. Qiu, L. Sha, S. P. Miller, and D. D. Cofer, "A formal architecture pattern for real-time distributed systems," in *RTSS'09*. IEEE, 2009.

[2] K. Bae, J. Meseguer, and P. C. Ölveczky, "Formal patterns for multirate distributed real-time systems," *Science of Computer Programming*, vol. 91, Part A, pp. 3 – 44, 2014.

[3] J. Meseguer and P. C. Ölveczky, "Formalization and correctness of the PALS architectural pattern for distributed real-time systems," *Theoretical Computer Science*, vol. 451, pp. 1–37, 2012.

[4] A. Al-Nayeem, L. Sha, D. D. Cofer, and S. P. Miller, "Pattern-based composition and analysis of virtually synchronized real-time distributed systems," in *ICCPS'12*. IEEE, 2012.

[5] K. Bae and P. C. Ölveczky, "Hybrid multirate PALS," in *Logic, Rewriting, and Concurrency*, ser. LNCS. Springer, 2015, to appear.

[6] S. Gao, J. Avigad, and E. M. Clarke, "δ-complete decision procedures for satisfiability over the reals," in *IJCAR'12*, ser. LNCS, vol. 7364. Springer, 2012.

[7] S. Gao, S. Kong, and E. M. Clarke, "Satisfiability modulo ODEs," in *FMCAD'13*. IEEE, 2013.

[8] ——, "dReal: An SMT solver for nonlinear theories over the reals," in *CADE'13*, ser. LNCS, vol. 7898. Springer, 2013.

[9] K. Bae, J. Krisiloff, J. Meseguer, and P. C. Ölveczky, "Designing and verifying distributed cyber-physical systems using Multirate PALS: An airplane turning control system case study," *Science of Computer Programming*, vol. 103, pp. 13–50, 2015.

[10] B. L. Stevens and F. L. Lewis, *Aircraft control and simulation*. John Wiley & Sons, 2003.

[11] S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preußig, M. Remelhe, S. Simon, and H. Treseler, "A case study in tool-aided analysis of discretely controlled continuous systems: The two tanks problem," in *Hybrid Systems V*, ser. LNCS. Springer, 1999, vol. 1567.

[12] J. Raisch, E. Klein, S. O'Young, C. Meder, and A. Itigin, "Approximating automata and discrete control for continuous systems — two examples from process control," in *Hybrid Systems V*, ser. LNCS. Springer, 1999, vol. 1567.

[13] S. Ratschan and Z. She, "Safety verification of hybrid systems by constraint propagation-based abstraction refinement," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 1, 2007.

[14] M. Fränzle and C. Herde, "HySAT: An efficient proof engine for bounded model checking of hybrid systems," *Formal Methods in System Design*, vol. 30, no. 3, pp. 179–198, 2007.

[15] A. Eggers, M. Fränzle, and C. Herde, "SAT modulo ODE: A direct SAT approach to hybrid systems," in *ATVA'08*, ser. LNCS. Springer, 2008, vol. 5311.

[16] A. Cimatti, S. Mover, and S. Tonetta, "SMT-based verification of hybrid systems," in *26th AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

[17] ——, "A quantifier-free SMT encoding of non-linear hybrid automata," in *FMCAD'12*. IEEE, 2012.

[18] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *CAV'11*, ser. LNCS, vol. 6806. Springer, 2011.

[19] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *CAV'13*, ser. LNCS, vol. 8044. Springer, 2013.