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*—For effectively designing and analyzing *virtually synchronous* cyber physical systems, the PALS methodology has been proposed to reduce the combinatorial complexity caused by the system's concurrency. However, cyber physical systems are often *distributed hybrid systems* in which distributed controllers communicate with each other via a network and govern physical entities with continuous dynamics. This paper presents Hybrid PALS to reduce the design and analysis of virtually synchronous hybrid systems to those of the underlying synchronous model, and shows a bisimulation equivalence between a synchronous model and a distributed hybrid model. We explain how various formal analysis problems of hybrid PALS models, such as bounded reachability and inductive analysis, can be reduced to SMT solving over the real numbers with nonlinear ordinary differential equations up to an arbitrary precision. Since such SMT-based analysis of distributed hybrid systems are typically unfeasible due to the *formula explosion problem*, we also proposes a novel SMT framework to effectively encode distributed hybrid systems in a modular way. Our techniques greatly increase the performance of SMT analysis for nontrivial distributed CPS.**

I. Introduction

Virtually synchronous cyber physical systems (CPS) consist of a number of distributed controllers that should logically act together in a synchronous way. The devices may operate at different frequencies, but the communication happens at their hyper-period boundary (e.g., avionics, aerospace, automotive, robotics, etc). The design and analysis of such systems is very hard because of race conditions, clock skews, network delays, and execution times. For this reason, the PALS (physically asynchronous logically synchronous) methodology [1]–[4] has been developed to reduce the system complexity of a virtually synchronous CPS, provided that the network infrastructure gives *performance bounds* on computation times, network delays, and imprecisions of the local clocks.

In this paper we present *Hybrid PALS* that extends the PALS methodology to virtually synchronous CPS given by *distributed hybrid systems*. The devices also interact with their local *physical environments*, with continuous dynamics specified by using ordinary differential equations (ODE). We show that there exists a *bisimulation equivalence* relating distributed hybrid systems and their underlying synchronous models. This means that the design and analysis of distributed hybrid systems can be reduced to those of the synchronous models. Hybrid PALS was first proposed in [8], but the work imposes strong stringent conditions on sampling and response timings of sensors and actuators. Moreover, [8] does not show a bisimulation equivalence (but only a trace equivalence).

However, formal analysis of virtually synchronous CPS with continuous dynamics is still very difficult. Inputs or parameters of CPS can be any real numbers in a certain bound. Different components will read their physical values at possibly different times according to their local clocks. The continuous dynamics of CPS often involves *nonlinear* ODEs which may not provide exact solutions in general. Although the PALS methodology can greatly reduce the analysis effort caused by distributed real-time systems, the difficulties raised by distributed hybrid systems cannot be removed without using low-quality approximation on ODEs and local clocks.

We present symbolic techniques using SMT solving to analyze distributed CPS along with PALS. Formal analysis problems of distributed CPS, involving nonlinear ODEs and clock skews, are reduced to the satisfaction of SMT formulas over the real numbers and ODEs. The satisfiability problem (generally undecidable) is decidable up to any given precision $\delta > 0$ [5], [6]. The number δ, provided by the user, is the bound on numerical errors that is tolerable in the analysis. By this approach, we can apply state-of-the-art SMT techniques that are effective for analyzing *discrete* controllers, while precisely analyzing the continuous dynamics of CPS in a robust way under I/O sampling or timing jitters by δ.

One obstacle to SMT-based methods is that it is not scalable 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 separately encoded in SMT, so that ODEs are considered *only after* the discrete part of the system is fully analyzed. Second, inductive SMT analysis, besides bounded reachability, is used to verify safety properties of distributed CPS for unbounded time. Third, compositional SMT analysis is used to analysis each part of distributed CPS by a divided-and-conquer approach.

Another obstacle is that SMT-based analysis is currently inefficient for *distributed* hybrid systems with nonlinear ODEs. In distributed CPS, physical states of different components can be physically correlated to each other. For example, if we consider two adjacent rooms with thermostat controllers, the temperature of one room immediately affects the temperature of the other room. The behavior is specified as *coupled* ODEs where variables evolve simultaneously over continuous time. Existing SMT approaches use the non-modular encoding for such distributed hybrid systems. The size of the SMT formula can be huge, and this leads to the *formula explosion problem* that makes such SMT-based analysis practically infeasible.

For this reason, we present a novel 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. The satisfaction problems of the new logical theory can be reduced to ones in the standard theory of the real numbers *at no cost*. We have implemented our SMT technique within the dReal SMT solver [7]. The experimental results show that our techniques can dramatically increase the performance of SMT-based analysis for nontrivial distributed CPS.

The rest of the paper is organized as follows. Section II briefly recalls PALS. Section III explains how PALS can be applied to distributed hybrid systems. Section IV shows how PALS models for distributed CPS and their analysis problems (such as bounded reachability, inductive, and compositional analysis) can be symbolically encoded as logic formulas. Section V presents a new SMT framework that allows modular encoding for distributed hybrid systems. Section VI presents case studies and their verification using our methods based on PALS and SMT. Finally, Section VII explains some related work, and Section VIII presents some concluding remarks.

II. PALS Overview

PALS transforms a *synchronous design* SD with global period T into a *correct-by-construction* distributed real-time system $\mathcal{MA}(SD, T, \Gamma)$, provided that the underlying network infrastructure give 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, executing a transition, and real-time scheduling 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. For each iteration, a machine performs a transition based on its current state and the inputs (from other machines), 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 (a value from the* j*-th* output port *is an element of* D_{o_j}*), 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* connects the input and output ports of the machines or the ensemble $\mathcal{E}$, where no connection exists between two fast machines.

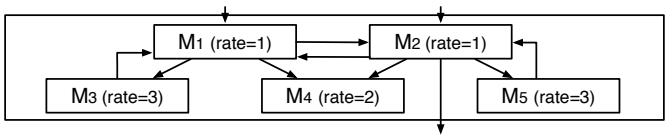

Fig. 1. A multirate ensemble $\mathcal{E}$, with M_1 and M_2 slow machines.

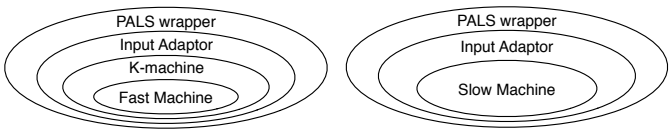

Fig. 2. The wrapper hierarchies in PALS distributed real-time models.

In each iteration, all components in $\mathcal{E}$ perform the transitions *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 applied 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.

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 $\mathcal{MA}(\mathcal{E}, T, \Gamma)$, an innermost machine performs at its own rate according to its local clock that deviates by less than ϵ from the global 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, some timers, and access to the machine's local clock. A PALS wrapper shares the same *global period* T. Received inputs are stored in the input buffer. When a new 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 all the inner components is finished *and* the backoff timer expires (before $u_0 + \max(2\epsilon - \mu_{min}, \alpha_{max})$), the contents of the output buffer are sent out, and delivered to the destination before $u = \mu_{max} + u_0 + \max(2\epsilon - \mu_{min}, \alpha_{max})$. 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 $u < (i+1)T - \epsilon$.

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.

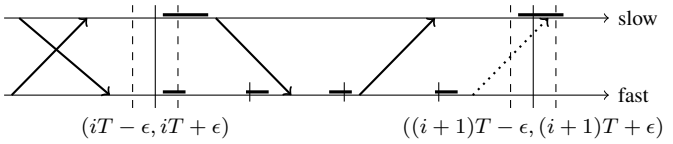

Fig. 3. Timeline for $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ ($k = 4$ and $k' = 3$). Diagonal arrows denote network transmission and short horizontal lines denote the execution. The dotted arrow illustrates that the outputs can arrive after the beginning of the next round if the fast machine waits until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

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 $k' < k$, 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 ignores 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 a distributed real-time model $\mathcal{MA}(\mathcal{E}, T, \Gamma)$, network transmission can happen only in between $(iT+\epsilon, (i+1)T-\epsilon)$ 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 $M_{\mathcal{E}}$. There exists a function *sync* associating each stable state with the corresponding state of $M_{\mathcal{E}}$. 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. PALS for Distributed Hybrid Systems

This section presents *Hybrid PALS* to incorporate physical environments in PALS, introduced in [8]. A state of a physical environment is given by 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 using 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 considered as just nondeterministic models defined for any possible environment behavior. Then, the *environment restrictions* $\mathcal{E} \restriction E$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction E$ define the behavior of the models constrained by a specific physical environment E. In this section we also present a bisimulation equivalence relating $\mathcal{E} \restriction E$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction E$, which generalizes the previous trace equivalence result in [8].

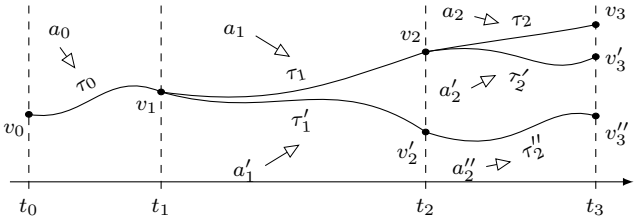

Fig. 4. A controlled physical environment E_M.

A. Controlled Physical Environments

The physical environment E_M is specified as a *controlled physical environment* that defines every possible trajectory of its physical parameters $\vec{x}$ 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 controlled 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* 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 can be physically correlated to each other, and one environment may immediately affect another environment. Such physical correlations are expressed as *time-invariant constraints* $(\forall t.\, \psi)$ of physical parameters over time t. For instance, 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 expressed as $\forall t.\, x_1(t) = x_2(t)$.

B. Environment-Restricted Controllers

A controller M is a *nondeterministic* machine parameterized by any behavior of its environment E_M. Since the controller M interacts with E_M according to its local clock, we take into account the local clock of M. The local clock may differ from the global clock 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* $M \restriction E_M$ 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 $iT - \epsilon$. After some sampling time t_I has elapsed, 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 machine state s. After the execution finished at time $c_M(i) + t_R$, its machine state changes to the next one s', where t_R denotes the execution time. The new outputs $\vec{o}$ from M are delivered to their destination before $(i+1)T - \epsilon$.

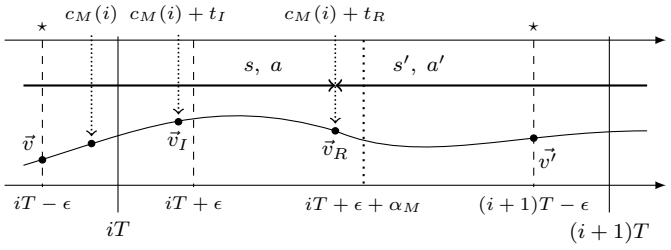

Fig. 5. Timeline for an environment-restricted controller $M \restriction E_M$ with a local clock c_M, an environment sampling time t_I, and an response time t_R.

A control command from a controller M depends on M's current machine state. In Fig. 5, a current control command a (given by s) remains effective until the execution finished at time $c_M(i) + t_R$, and then a new control command a' (given by s') 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*.

Moreover, $M \restriction E_M$ can be considered as a "normal" state machine in *synchronous models*, provided that the timing values (such as $c_M(i)$, t_I, and t_R) as well as control commands are all determined by M's machine states and inputs.

Definition 3. *Suppose that M has a state space of the form $S \times \mathbb{R}^m$ for $\mathbb{R}^m$ the m physical parameters that M observes. For $(s, \vec{v}_o) \in S \times \mathbb{R}^m$, the* interface *is given by the projection functions: (i) $\pi_T(s) \in \mathbb{N}$ a "round" number; (ii) $\pi_R(s, \vec{i}) \in \mathbb{R}$ an environment response time; (iii) $\pi_I(s) \in \mathbb{R}$ an environment 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 $\pi_O(\vec{v}) \in \mathbb{R}^m$.*

Given state $(s, \vec{v})$ and input $\vec{i}$, the next state $(s', \vec{v'})$ and output $\vec{o}$ can be constructed by the same process of Fig. 5:

Definition 4. *The* environment restriction *of M by E_M is the machine $M \restriction E_M = (D_i, S \times \mathbb{R}^l, D_o, \delta_{M\restriction E_M})$, where $((\vec{i}, (s, \vec{v})), ((s', \vec{v'}), \vec{o})) \in \delta_{M\restriction E_M}$ 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 discrete controllers (in $\mathcal{E}$ or $\mathcal{MA}(\mathcal{E}, T, \Gamma)$), their physical environments, and timing policies.

Deciding π is one of system design choices. Most generally, values of π can vary according to states and inputs. More realistically, sampling/response timings are fixed to certain numbers to simplify the system complicity. For example, in our previous work [8], t_I is always 0 (for controllers tightly integrated with its environments) and t_R is always "delayed" for 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 $M_j \restriction E_{M_j}$ 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 lock-step composition of such transitions *that also satisfy 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 $M_j \restriction E_{M_j}$ 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 $(\sim_{obi}; \mathit{sync})$ in Theorem 1). Consequently, we have:

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

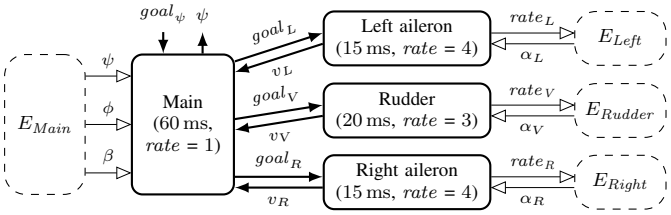

Fig. 6. The distributed controllers for turning an airplane.

D. Example: Turning an Airplane

We consider a distributed hierarchical CPS to control the turning maneuver of an airplane (adapted from [9]). To make a turn, an aircraft rolls towards the direction of the turn by moving its ailerons (surfaces attached to the end of the wings). The rolling causes a yawing moment in the opposite direction, called *adverse yaw*, 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 coordinated turn.

Fig. 6 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}. For each period, 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}. The continuous dynamics of α_M is given 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. For each period, 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 values of the direction angle ψ, the roll angle ϕ, and the yaw angle β.

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]:

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

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 $\forall t.\,(\delta_L(t) = \alpha_L(t)) \wedge (\delta_V(t) = \alpha_V(t)) \wedge (\delta_R(t) = \alpha_R(t))$. The PALS models consist of the multirate ensemble $\mathcal{E}$, the physical environments, and the time-invariant constraint, along with appropriate sampling and response timing policies Π.

IV. Logical Encoding of Hybrid PALS Models

Thanks to the behavioral equivalence (Theorem 2), analysis problems of a distributed CPS $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are reduced to those on the simpler synchronous model $\mathcal{E} \restriction_\Pi E_\mathcal{E}$. However, it is difficult to analyze $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ with (nonlinear) continuous dynamics and skews of the local clocks. The concrete values of local clocks are unknown since the values are determined on-the-fly by *clock synchronization* mechanisms [1], [11]. Numerical simulation can no longer be used for nonlinear dynamics because numerical errors can be accumulated.

This section explains how analysis problems of $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ for *any possible local clock* can be symbolically encoded as logic 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 $\delta > 0$ (see Section V).

A. Representing Environment-Restricted Controllers

A controller M is naturally expressed as a logic formula over discrete domain of the form $\phi_M(\vec{i}, \vec{y} \shortmid \vec{y'}, \vec{o})$, including *variables* $\vec{i}$ denoting input, $\vec{y}$ denoting the current state, y' denoting the next state, and $\vec{o}$ denoting 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 is expressed as a formula over the real numbers of the form $\phi_{E_M}(\vec{a}, u_0, u_t, \vec{v} \shortmid \vec{x})$, including: (i) *unary function symbols* $\vec{x}$ denoting the trajectories of E_M's physical parameters $\vec{x}$, and (ii) *variables* $\vec{a}$ denoting control commands, u_0 and u_t respectively denoting times at the beginning and the end of the trajectory duration, and $\vec{v}$ denoting 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 $\vec{a}$, then the formula ϕ_{E_M} includes universal quantification over time along with solutions of the ODEs, such as a formula of the form:

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

For example, the formula $\phi_{E_M}(rate_M, u_0, u_t, v_M \shortmid \alpha_M)$ for E_M of a subcontroller M in the airplane example is given by: $\forall t \in [u_0, u_t].\ \alpha_M(t) = v_M + \int_0^{t-u_0} rate_M\,\mathrm{d}t$.

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 $t \in [u_0, u_t]$.*

For an environment restriction $M \restriction E_M$, its formula $\phi_{M\restriction E_M}$ has the form $\phi^{T,i}_{M\restriction E_M}(\vec{i}, \vec{y}, \vec{v} \shortmid \vec{y'}, \vec{v'}, \vec{o})$, including 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 explicitly writing $iT - \epsilon$ and $(i+1)T - \epsilon$ in $\phi_{M\restriction E_M}$, we use the "ϵ-shifted" time axis for the interval $[iT, (i+1)T]$. But 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 unpredictable. Because $iT - \epsilon < c_M(i) < iT + \epsilon$, we represent those times as formulas $t_I < u_I < t_I + 2\epsilon$ and $t_R < u_R < t_R + 2\epsilon$ within $\phi_{M\restriction E_M}$.

Definition 8. *For an environment restriction $M \restriction E_M$ with the interface projection functions $\pi = (\pi_T, \pi_R, \pi_I, \pi_C, \pi_O)$, $\phi^{T,i}_{M\restriction E_M}(\vec{i}, \vec{y}, \vec{v} \mid \vec{y'}, \vec{v'}, \vec{o})$ is defined from Definition 4 by:*

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

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

B. Representing Hybrid Ensembles

A synchronous composition of $\mathcal{E}$ is a single machine $M_{\mathcal{E}}$ in which all machines perform their transitions in lock step. Each fast machine f performs $k = rate(f)$ internal transitions in one step. A formula $\phi^{T,i}_{(M\restriction E_M)^{\times k}}$ for the k-step *deceleration* of an environment restriction $M \restriction E_M$ is defined by sequentially composing k subformulas $\phi^{T/k,ik}_{M\restriction E_M}, \ldots, \phi^{T/k,(ik+k-1)}_{M\restriction E_M}$. These formulas are respectively related to the k equal *subintervals* $[(ik+n-1)T/k - \epsilon, (ik+n)T/k - \epsilon]$ for $n = 1, \ldots, k$.

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

$$\exists \{\vec{y_n}, \vec{v_n}\}_{n=1}^{k-1}. \bigwedge_{n=1}^{k} \phi^{T/k,(ik+n-1)}_{M\restriction E_M}(\vec{i_n}, \vec{y}_{n-1}, \vec{v}_{n-1} \mid \vec{y_n}, \vec{v_n}, \vec{o_n})$$

The wiring diagram of $\mathcal{E}$ is expressed by 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.*

For a hybrid ensemble $\mathcal{E} \restriction_\Pi E_{\mathcal{E}}$ of machines $\{M_j\}_{j\in J_S\cup J_F}$ and their physical environments $\{E_{M_j}\}_{j\in J_S\cup J_F}$ (J_S denoting slow machines and J_F denoting fast machines), its formula has the form $\phi^{T,i}_{\mathcal{E}\restriction_\Pi E_{\mathcal{E}}}(\vec{i}, \{\vec{z_j}, \vec{f_j}\}_{j\in J_S\cup J_F} \mid \{\vec{z'_j}, \vec{f'_j}\}_{j\in J_S\cup J_F}, \vec{o})$, including variables: (i) $\vec{i}$ denoting input, (ii) $\vec{z_j}$ denoting a state of $M_j \restriction E_{M_j}$ at the beginning of the round (at time $iT-\epsilon$), (iii) $\vec{f_j}$ denoting feedback outputs from the previous round, (iv) $\vec{z'_j}$ denoting a state of $M_j \restriction E_{M_j}$ at the end of the round (at time $(i+1)T-\epsilon$), (v) $\vec{f'_j}$ denoting the feedback outputs for the next round, and (vi) $\vec{o}$ denoting output.

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

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

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. Representing Formal Analysis 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 $\forall t.safe(\vec{z}, t)$ with respect to state variables $\vec{z}$ and time variable t.

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

Definition 12. *A bounded analysis problem for $\forall t.safe(\vec{z}, t)$ up to n with an initial condition $init(\vec{z})$ is expressed by:*

$$\begin{aligned}
&\exists \vec{z}_0, \{\vec{z_k}, \vec{i_k}, \vec{o_k}, t_k\}_{k=1}^n. \textstyle\bigwedge_{k=1}^n \left(\phi^{T,k}_{\mathcal{E}\restriction_\Pi E_{\mathcal{E}}}(\vec{i_k}, \vec{z}_{k-1} \mid \vec{z_k}, \vec{o_k})\right) \\
&\wedge init(\vec{z}_0) \ \wedge \ \textstyle\bigvee_{k=1}^n \left((k-1)T < t_k \le T \wedge \neg safe(\vec{z_k}, t_k)\right).
\end{aligned}$$

For unbounded verification of a safety property, we encode its inductive proof as logic formulas. 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. *An inductive analysis problem for $\forall t.safe(\vec{z}, t)$ with an inductive condition $ind(\vec{z})$ is expressed by:*

$$\begin{aligned}
&\forall \vec{z}, \vec{z'}, \vec{i}, \vec{o}, t \in [0, T]. \left(ind(\vec{z}) \to safe(\vec{z}, t)\right) \\
&\qquad \wedge \left((ind(\vec{z}) \ \wedge \ \phi^{T,0}_{\mathcal{E}\restriction_\Pi E_{\mathcal{E}}}(\vec{i}, \vec{z} \mid \vec{z'}, \vec{o})) \to ind(\vec{z'})\right).
\end{aligned}$$

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. *A 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 expressed by:*

$$\begin{aligned}
&\forall \vec{z}, \vec{z'}, \vec{i}, \vec{o}, t \in [0, T]. \left(c_{in}(\vec{i}, \vec{z}, t) \to safe(\vec{z}, t)\right) \\
&\wedge \left((c_{in}(\vec{i}, \vec{z}, t) \ \wedge \ \phi^{T,0}_{\mathcal{E}\restriction_\Pi E_{\mathcal{E}}}(\vec{i}, \vec{z} \mid \vec{z'}, \vec{o})) \to c_{out}(\vec{o}, \vec{z'}, t)\right).
\end{aligned}$$

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 CPS

Various formal analysis problems of distributed CPSs can be encoded as logic formulas over the real numbers and ODEs. The satisfiability of those formulas is undecidable for nonlinear hybrid systems in general, but becomes *decidable* if we take account of robustness properties under numerical perturbations $\delta > 0$. 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. This is practically very useful since sampling exact values of physical parameters is not possible in reality.

However, formulas for hybrid PALS models may contain ODEs and universal quantification over *uninterpreted real functions*, which are not supported by current state-of-the-art SMT solving techniques. For example, the ODEs in E_{Main} in the airplane example include uninterpreted function symbols δ_L, δ_V, and δ_R for control angles. The correspondences between $(\delta_L, \delta_V, \delta_R)$ and the surface angles $(\alpha_L, \alpha_V, \alpha_R)$ of the subcontrollers are given by the time-invariant constraint $\forall t.\, (\delta_L(t) = \alpha_L(t)) \wedge (\delta_V(t) = \alpha_V(t)) \wedge (\delta_R(t) = \alpha_R(t))$.

This section shows how formulas for hybrid PALS models (presented in Section IV) can be equivalently encoded as *SMT formulas* without the use of universal quantification over uninterpreted real functions. For this purpose, we present a new SMT framework to provide an efficient SMT algorithm for formulas generated from hybrid PALS models.

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, $\mathcal{L}_{\mathcal{F}} = (\mathcal{F}, >)$ 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 as *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 express logic formulas for hybrid PALS models in $\mathcal{L}_{\mathcal{F}}$, which typically involve different local physical environments interacting with each other. Therefore, we 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 much 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 the following *named* real functions:

$$\begin{aligned} \mathtt{m}_1 \quad & \sin(x(t)) : Real \to Real, \\ \mathtt{m}_2 \quad & [y(t)+1, z(t)^2] : Real \to Real^2. \end{aligned}$$

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

$$\begin{aligned} int^2(\mathtt{m}_2, 1) &\equiv \int_0^1 [y+1, z^2] \mathrm{d}t, \\ int^{1,2}(\mathtt{m}_1, \mathtt{m}_2, T) &\equiv \int_0^T [\sin(x), y+1, z^2] \mathrm{d}t. \end{aligned}$$

Finally, 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.,

$$app^1(m_1, 3) \equiv \sin(x(3)), \quad app^2(m_2, u) \equiv [y(u)+1, z(u)^2].$$

These operators ensure that each name object is really related to a concrete unary real function with domain $\mathbb{R}$.

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 a $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ 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 the *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.

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 show how hybrid PALS models are encoded in the new SMT logic $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ without the use of uninterpreted real functions and universal quantification. We restrict our attention to time-invariant constraints with only equality terms, such as one for the airplane example, since physical correlations for CPS 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 another, e.g., by replacing x_1 with x_2. From now on we assume that time-invariant constraints are already removed from the formula by this process.

Now here a basic idea: uninterpreted function by name, and universal quantification by equality

The formula $\phi^{T,0}_{M_j \restriction E_{M_j}}$ for the first step of each environment restriction $M_j \restriction E_{M_j}$ in a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ involves the following subformula for sampling time u^j_I and response time u^j_R:

$$\begin{aligned}
&\forall t \in [0, u^j_I].\ \vec{x}_j(t) = \vec{v}_j + \textstyle\int_0^t F_j(\vec{x}_j, t)\,\mathrm{d}t \ \wedge \\
&\forall t \in [u^j_I, u^j_R].\ \vec{x}_j(t) = \vec{x}_j(u_I) + \textstyle\int_0^{t-u_I} F_j(\vec{x}_j, t)\,\mathrm{d}t \ \wedge \\
&\forall t \in [u^j_R, T].\ \vec{x}_j(t) = \vec{x}_j(u_R) + \textstyle\int_0^{t-u_R} F_j(\vec{x}_j, t)\,\mathrm{d}t \ \wedge \\
&u^j_I \in [t^j_I, t^j_I + 2\epsilon] \ \wedge\ u^j_R \in [t^j_R, t^j_R + 2\epsilon]
\end{aligned}$$

For networks of hybrid systems, the standard SMT encoding tends to yield a formula including many ODE literals. For a parallel composition $H_1 \parallel \ldots \parallel H_n$, the formula for k-step bounded model checking has $N = (k+1) \cdot \prod_{i=1\ldots n} m_i$ ODE literals, where each H_i has m_i modes. The size of the formula is $O(k \cdot \prod_{i=1}^n m_i)$.

Therefore, SMT-based analysis for networks of hybrid systems can easily suffer from the *formula explosion problem* that makes such analysis practically infeasible.

The basic idea is to decompose a single ODE literal into a conjunction of several literals according to its underlying structure. Consider a l-dimensional ODE literal $\vec{y}^t = \vec{y}^0 + \int_0^T \mathit{flow}(\vec{x})\,\mathrm{d}t$. Mathematically, a variable $x \in \vec{x}$ in $\mathit{flow}(\vec{x})$ can be considered a unary function $\mathbb{R} \to \mathbb{R}$, and thus the term $\mathit{flow}(\vec{x})$ can also be considered as a unary function $\mathbb{R} \to \mathbb{R}^l$. Therefore, using an extra function symbol $g : \mathbb{R} \to \mathbb{R}^l$ and the state function symbols $\vec{x}(t)$, the ODE literal can be rewritten as the logically equivalent conjunction:

$$\vec{y}^t = \vec{y}^0 + \textstyle\int_0^T g(t)\,\mathrm{d}t \quad \wedge \quad \forall t \in [0, T].\ g(t) = \mathit{flow}(\vec{x})(t).$$

The above technique is particularly useful for a parallel composition $H_1 \parallel \ldots \parallel H_n$, since an ODE literal can be decomposed with respect to its components $H_1, \ldots, H_n$. Let $\overrightarrow{\vec{x}} = [\vec{x}_1, \ldots, \vec{x}_n]$ $\overrightarrow{\vec{x}}^t = [\vec{x}^t_1, \ldots, \vec{x}^t_n]$ and $\overrightarrow{\vec{x}}^0 = [\vec{x}^0_1, \ldots, \vec{x}^0_n]$ denote n-vectors of state variable vectors. Consider a single *concrete* ODE literal

$$\overrightarrow{\vec{x}}^t = \overrightarrow{\vec{x}}^0 + \int_0^T [\mathit{flow}_1(\overrightarrow{\vec{x}}), \ldots, \mathit{flow}_n(\overrightarrow{\vec{x}})]\,\mathrm{d}t.$$

By using n extra function symbols $g_1, \ldots, g_n$ that respectively correspond to $H_1, \ldots, H_n$, we have a logically equivalent conjunction

$$\begin{aligned}
\overrightarrow{\vec{x}}^t = \overrightarrow{\vec{x}}^0 + \int_0^T [g_1(t), \ldots, g_n(t)]\,\mathrm{d}t \ \wedge \\
\bigwedge_{i=1}^n \forall t \in [0, T].\ g_i(t) = \mathit{flow}_i(\overrightarrow{\vec{x}})(t).
\end{aligned}$$

In a way similar to the above, the composed continuous behavior for duration T with respect to arbitrary flows $\overrightarrow{g} = [g_1, \ldots, g_n]$ in $(q_1, \ldots, q_n) \in Q_1 \times \cdots \times Q_n$ from values $\overrightarrow{\vec{x}}^0$ to $\overrightarrow{\vec{x}}^t$ is then expressed as the formula:

$$\begin{aligned}
\mathit{cont}_{\overrightarrow{g}}(\overrightarrow{\vec{x}}^0, \overrightarrow{\vec{x}}^t, \overrightarrow{m}, T) \ \equiv\ \overrightarrow{\vec{x}}^t = \overrightarrow{\vec{x}}^0 + \int_0^T [g_1(t), \ldots, g_n(t)]\,\mathrm{d}t \ \wedge \\
\forall U \in [0, T]\ \forall \overrightarrow{\vec{x}}^u \in \mathit{val}(X_1 \cup \cdots \cup X_n). \\
\overrightarrow{\vec{x}}^u = \overrightarrow{\vec{x}}^0 + \int_0^U [g_1(t), \ldots, g_n(t)]\,\mathrm{d}t \ \to\ \mathit{inv}(\overrightarrow{m}, \overrightarrow{\vec{x}}^u).
\end{aligned}$$

SMT formulas for hybrid systems by the new encoding can be rewritten in this logic without using uninterpreted real functions and universal quantification. Each extra function symbol g_i is replaced by a name variable w_i, and each subformula of the form $(\forall t \in [0, T_i].\ g_i(t) = \mathit{flow}_q(\vec{x})(t))$ is replaced by an equality $w_i = \mathtt{flow}_q$, where name *constant* $\mathtt{flow}_q$ denotes the flow function $\mathit{flow}_q(\vec{x})(t)$. For example, the continuous behavior of H with respect to name variable w_i is expressed as the formula:

$$\begin{aligned}
\mathit{cont}(w_i, \vec{x}^0_i, \vec{x}^t_i, m_i, T_i) \ \equiv\ (\vec{x}^t_i = \vec{x}^0_i + \mathit{int}^l(w_i, T_i)) \wedge \\
\forall U \in [0, T]\ \forall \vec{x}^u \in \mathit{val}(X).\ (\vec{x}^u_i = \vec{x}^0_i + \mathit{int}^l(w_i, U)) \\
\to\ \mathit{inv}(m_i, \vec{x}^u_i),
\end{aligned}$$

and the formula for k-step bounded model checking of H is rewritten as the following formula including equality terms $w_0 = \mathtt{flow}_q$ and $w_i = \mathtt{flow}_{q'}$:

$$\begin{aligned}
&\exists m_0, \ldots, m_k\ w_0, \ldots, w_k\ \vec{x}_0, \ldots, \vec{x}^t_k \in \mathit{val}(X)\ T_0, \ldots, T_k \in [0, T_{\max}]. \\
&\qquad \mathit{cont}(w_0, \vec{x}^0_0, \vec{x}^t_0, m_0, T_0) \wedge \\
&\qquad \textstyle\bigvee_{q \in Q} (m_0 = q \wedge \mathit{init}_q(\vec{x}^0_0) \wedge w_0 = \mathtt{flow}_q) \wedge \\
&\qquad \bigwedge_{i=1}^k \mathit{cont}(w_i, \vec{x}^0_i, \vec{x}^t_i, m_i, T_i) \wedge \\
&\qquad \bigvee_{q, q' \in Q} \begin{pmatrix} m_{i-1} = q \wedge m_i = q' \wedge \\ \mathit{jump}_{q,q'}(\vec{x}^t_{i-1}, \vec{x}^0_i) \wedge w_i = \mathtt{flow}_{q'} \end{pmatrix} \wedge \neg \mathit{safe}(\vec{x}^t_k)
\end{aligned}$$

Theorem 5. *Formulas (**??**–**??**) for formal analysis of hybrid systems rewritten in $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ are equivalent to the corresponding formulas in the standard encoding.*

Proof. This immediately follows from $\mathbb{R}_\mathcal{F} \subseteq \mathbb{R}_{\mathcal{F}\cup\mathcal{N}}$ and the formula rewriting process in Section **??**. Consider the above $\mathcal{L}_{\mathcal{F}\cup\mathcal{N}}$ formula for k-step bounded model checking. For the initial step, by distributing $\mathit{cont}(w_0, \vec{x}^0_0, \vec{x}^t_0, m_0, T_0)$ over $\bigvee_{q \in Q}(m_0 = q \wedge \mathit{init}_q(\vec{x}^0_0) \wedge w_0 = \mathtt{flow}_q)$, we have the equivalent formula $\bigvee_{q \in Q}(m_0 = q \wedge \mathit{init}_q(\vec{x}^0_0) \wedge w_0 = \mathtt{flow}_q \wedge \mathit{cont}(w_0, \vec{x}^0_0, \vec{x}^t_0, m_0, T_0))$. By replacing w_0 by $\mathtt{flow}_q$, we have $\bigvee_{q \in Q}(m_0 = q \wedge \mathit{init}_q(\vec{x}^0_0) \wedge \mathit{cont}(\mathtt{flow}_q, \vec{x}^0_0, \vec{x}^t_0, m_0, T_0))$. Since the integral term $\mathit{int}^l(\mathtt{flow}_q, T_0)$ denotes the function $\int_0^{T_0} \mathit{flow}_q(\vec{x})\,\mathrm{d}t$ of T_0, $\mathit{cont}(\mathtt{flow}_q, \vec{x}^0_0, \vec{x}^t_0, m_0, T_0)$ is equivalent to $\mathit{cont}^q_{\vec{x}^0_0}(\vec{x}^0_0, \vec{x}^t_0, T_0)$. Therefore, the 0-step formula is equivalent to $\bigvee_{q \in Q}\left(m_0 = q \wedge \mathit{init}_q(\vec{x}^0_0) \wedge \mathit{cont}^q_{\vec{x}^0_0}(\vec{x}^0_0, \vec{x}^t_0, T_0)\right)$, i.e., the 0-step in Formula (**??**). The other cases are similar. □

VI. Case Studies

This section presents four case studies for SMT-based analysis of networks of hybrid systems. These case studies involve nontrivial (nonlinear) ODEs, because of continuous physical

interaction between components. We have implemented our techniques in the dReal SMT solver, and performed formal analysis for these three case studies using the tool. We have compared the performance of the new encoding with the standard encoding, and with the existing heuristics used in the dReach tool [12], which *explicitly enumerates* all paths in the mode graph of a hybrid automaton and generates many small SMT formulas. All the experiments were conducted on Intel Xeon 2.0 GHz with 64 GB memory.

We have applied our techniques to design and verify two multirate distributed cyber physical systems: a distributed control system for turning an airplane where its effectors located in the airplane's wings (Section VI-A), and a robotics system with a conveyer belt and robot arms for physically simulating a Turing machine (Section **??**). Both systems are distributed *hybrid* systems consisting of *discrete* multirate periodic controllers, interconnected by a wired network, and interacting with their *continuous* local environments. The systems can provide PALS bounds Γ; in particular, a bound $\epsilon > 0$ on the imprecisions of the local clocks. Therefore, using Multirate PALS, the discrete behavior can be precisely modeled as a multirate synchronous system $\mathcal{E}$. However, the continuous behavior still depends on the exact values of the local clocks, given by clock synchronization mechanisms. The local physical environments of different components can be simultaneously correlated to each other, which cannot be modeled as network communications in general. The hybrid system modeling framework in Section **??** was shown to be effective for dealing with these difficulties. The bounded model checking method in Section **??** was applied to verify the given safety properties of the systems, with respect to (bounded) real number parameters and inputs, by means of `dReal`-based SMT solving in Section **??**.

A. Turning an Airplane

..

(adapted from [9]),

We consider a networked control system to perform the turning maneuver of an airplane, adapted from [9], in which a main controller directs subcontrollers for the ailerons and the rudder. To make a turn, an aircraft rolls by moving its ailerons (surfaces attached to the end of the wings), while its rudder (a surface attached to the vertical tail) is used to counter adverse yaw (a yawing moment caused by the rolling). The lateral dynamics of an aircraft can be specified as nonlinear differential equations [10]:

$$\dot{\beta} = Y(\beta, \delta_a, \delta_r)/mV - r + V/g * \cos(\beta) * \sin(\phi)$$
$$\dot{\phi} = p$$
$$\dot{p} = (c_1 r + c_2 p) \cdot r \cdot \tan(\phi) + c_3 L(\beta, \delta_a, \delta_r) + c_4 N(\beta, \delta_a, \delta_r)$$
$$\dot{\psi} = g/V \tan(\phi)$$
$$\dot{r} = (c_8 p - c_2 r) \cdot r \cdot \tan(\phi) + c_4 L(\beta, \delta_a, \delta_r) + c_9 N(\beta, \delta_a, \delta_r)$$

with: (i) state variables β for the yaw angle, p for the rolling moment, r for the yawing moment, ϕ for the roll angle, and ψ for the direction, (ii) control variables δ_a for the angle of the ailerons, and δ_r for the angle of the rudder, and (iii) linear functions $Y(\beta, \delta_a, \delta_r)$, $L(\beta, \delta_a, \delta_r)$, and $N(\beta, \delta_a, \delta_r)$ to denote side force, rolling moment, and yawing moment of the airplane. The main controller has one mode, and determines the desired control angles. The two subcontroller have two modes for moving up or down their surfaces, and gradually change the surface angles according to the main controller. We consider three variants of this model: a single-rate nonlinear model in which every controller has a 0.5 period, an approximated linear model assuming small perturbations given by linear differential equations [13], and a multirate linear model in which the aileron controller has a 0.25 period and the rudder controller has a $0.5/3$ period.

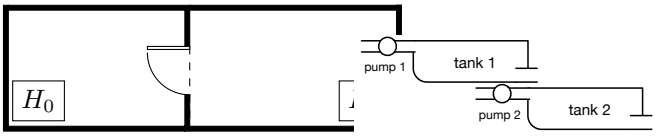

Fig. 7. Two interconnected rooms. Fig. 8. Two connected water tanks

B. Quadrotor Attitude Controller

..

C. Steam Boiler Controller

...

D. Networked Water Tank Controllers.

A number of water tanks are connected by pipes as shown in Figure 8. The water level in each tank is separately controlled by the pump in the tank (adapted from [14], [15]). Similarly, each water level depends on the pump's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the levels of the adjacent tanks. The water level x_i of tank i changes according to the differential equations:

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

where A_i, q_i, a are constants determined by the size of the tank, the power of the pump, and the width of the pipe, and g is the standard gravity constant ($x_0 = 0$ for the leftmost tank 1). There is also a shared timer variable τ with the flow condition $\dot{\tau} = 1$. Every pipe controller synchronously performs its discrete transitions: for each second, the pump is on if $x_i \leq L_{\min}$, and off if $x_i > L_{\max}$.

E. Networked Thermostat Controllers.

A network of classical thermostat hybrid automata is considered (where the specification of a single hybrid automaton is adapted from [16]). A number of rooms are interconnected by open doors, and the temperature of each room is separately controlled by each thermostat that turns its heater on and off. (e.g., Figure 7 for the case of two rooms). The temperature of each room depends on the heater's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the temperatures of the other rooms. E.g., for three rooms

$I = \{0, 1, 2\}$, the temperature x_i of room $i \in I$ changes according to the differential equations:

$$\dot{x}_i = K_i\big(h_i - ((1 - \textstyle\sum_{j\in I\setminus\{i\}} k_{i,j})x_0 + \sum_{j\in I\setminus\{i\}} k_{i,j}x_j)\big) \quad \text{if } m_i = m_{\texttt{on}},$$

$$\dot{x}_i = -K_i\big((1 - \textstyle\sum_{j\in I\setminus\{i\}} k_{i,j})x_0 + \sum_{j\in I\setminus\{i\}} k_{i,j}x_j\big) \quad \text{if } m_i = m_{\texttt{off}},$$

where $K_i, h_i \in \mathbb{R}$ are constants depending on the size of room i and the heater's power, respectively, and $k_{i,j} \in \mathbb{R}$ is determined by the size of the open door between rooms i and j. Every thermostat controller synchronously performs its discrete transitions. For each second, the heater is turned on if $x_i \leq T_{\min}$, and turned off if $x_i > T_{\max}$. To keep track of each one-second period, every automaton has a *shared* timer variable τ with the flow condition $\dot{\tau} = 1$.

F. Automated Highway System

...

VII. Related Work

hybrid automata techniques, not appropriate for hybrid PALS models, since they are basically time triggered models, which are not perfectly synchronized due to clock skews.

SMT-based approach is a relatively new direction for nonlinear hybrid system verification, because of the difficulty of handling SMT formulas over the reals with nonlinear functions. The research direction is initiated in the work [17], 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 [18], [19] and the MathSAT solver [20], [21]. But efficient encoding of networks of hybrid systems has not been investigated in existing work along these lines. On the other hand, for reachable set computation, SpaceEX [22] has involved methods for handling networks restricted to linear/multiaffine hybrid systems, and Flow* [23] proposed an approach for verifying nonlinear hybrid systems using Taylor model flowpipe construction. dReach [12] encodes reachability problems of hybrid systems to SMT problems and solves them using dReal [7]. Our approach is different from dReach, since dReach *explicitly* enumerates all mode paths of a hybrid automaton to generate many small SMT formulas for these paths.

Multirate PALS ?? (Abdullah's work, other synchronizer, virtually synchronization, ...)

Hybrid automata?

Approximation of hybrid systems?

SMT and `dReal` ??

$d\mathcal{L}$ and deductive methods ??

VIII. Concluding Remarks

..

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*. IEEE, 2009, pp. 161–170.

[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*. IEEE, 2012, pp. 65–74.

[5] S. Gao, J. Avigad, and E. M. Clarke, "δ-complete decision procedures for satisfiability over the reals," in *IJCAR*, ser. Lecture Notes in Computer Science, vol. 7364. Springer, 2012, pp. 286–300.

[6] S. Gao, S. Kong, and E. M. Clarke, "Satisfiability modulo odes," in *FMCAD*. IEEE, 2013, pp. 105–112.

[7] ——, "dReal: An SMT solver for nonlinear theories over the reals," in *CADE*, ser. Lecture Notes in Computer Science, vol. 7898. Springer, 2013, pp. 208–214.

[8] K. Bae and P. C. Ölveczky, "Hybrid multirate pals," in *Logic, Rewriting, and Concurrency*. Springer, 2015, to appear.

[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] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.

[12] S. Kong, S. Gao, W. Chen, and E. M. Clarke, "dReach: δ-reachability analysis for hybrid systems," in *TACAS*, ser. Lecture Notes in Computer Science, vol. 7898. Springer, 2015, pp. 200–205.

[13] D. Allerton, *Principles of flight simulation*. John Wiley & Sons, 2009.

[14] 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*. Springer, 1999, pp. 163–185.

[15] J. Raisch, E. Klein, C. Meder, A. Itigin, and S. O'Young, "Approximating automata and discrete control for continuous systems — two examples from process control," in *Hybrid systems V*. Springer, 1999, pp. 279–303.

[16] T. A. Henzinger, *The theory of hybrid automata*. Springer, 2000.

[17] 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, p. 8, 2007.

[18] 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. [Online]. Available: http://dx.doi.org/10.1007/s10703-006-0031-0

[19] A. Eggers, M. Fränzle, and C. Herde, "SAT modulo ODE: A direct sat approach to hybrid systems," in *Automated Technology for Verification and Analysis*. Springer, 2008, pp. 171–185.

[20] A. Cimatti, S. Mover, and S. Tonetta, "Smt-based verification of hybrid systems," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5072

[21] ——, "A quantifier-free SMT encoding of non-linear hybrid automata," in *Formal Methods in Computer-Aided Design, FMCAD 2012, Cambridge, UK, October 22-25, 2012*, 2012, pp. 187–195. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6462573

[22] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, 2011, pp. 379–395. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22110-1_30

[23] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*. Springer, 2013, pp. 258–263.

APPENDIX

Definition 17. *For a typed machine $M_j = (D_{i_j}, S_j, D_{o_j}, \delta_{M_j})$ with input set $D_{i_j} = D_{i_{j_1}} \times \cdots \times D_{i_{j_n}}$, an input adaptor $adap(j) = \{\alpha_i : D'_l \to D_{i_l}\}_{l \in \{1,\ldots,n_j\}}$ is a family of functions to give desired input values $(\alpha_1(d_1), \ldots, \alpha_n(d_n))$ from output $(d_1, \ldots, d_n) \in D_1 \times \cdots \times D_n$ of other machines.*

Definition 18. *A* multirate machine ensemble *is given by a tuple $\mathcal{E} = (J_S, J_F, e, \{M_j\}_{j \in J_S \cup J_F}, E, src, rate, adap)$ with: (i) J_S a set of slow machine indices; (ii) J_F a set of fast machine indices ($J_S \cap J_F = \emptyset$); (iii) $e \notin J_S \cup J_F$ the environment index; (iv) $\{M_j\}_{j \in J_S \cup J_F}$ a family of typed machines; (v) $E = (D_i^e, D_o^e)$ the* environment *with D_i^e the* input set *and D_o^e the* output set*; (vi) src a* wiring diagram *assigning to each input port its source output port, so that no connection exists between two "fast" machines; (vii) $rate$ assigning to a fast machine its* rate*; and (viii) $adap$ assigning to a machine its* input adaptor.

Definition 19. *The* synchronous composition *of an ensemble $\mathcal{E} = (J_S, J_F, e, \{M_j\}_{j \in J_s \cup J_F}, E, src, rate, adap)$ is the typed machine $M_{\mathcal{E}} = (D_i^{\mathcal{E}}, S^{\mathcal{E}}, D_o^{\mathcal{E}}, \delta_{\mathcal{E}})$ where: (i) $D_i^{\mathcal{E}} = D_o^e$ and $D_o^{\mathcal{E}} = D_i^e$; (ii) $S^{\mathcal{E}} = (\Pi_{j \in J} S_j) \times (\Pi_{j \in J} D_{OF}^j)$, where D_{OF}^j stores the "feedback outputs" of subcomponents; and (iii) $\delta_{\mathcal{E}} \subseteq (D_i^{\mathcal{E}} \times S^{\mathcal{E}}) \times (S^{\mathcal{E}} \times D_o^{\mathcal{E}})$ "combines" the transitions of the subcomponents into a synchronous step (see [3]).*

Definition 20. *The transition system for an ensemble $\mathcal{E}$ is a pair $ts(\mathcal{E}) = (S^{\mathcal{E}} \times \mathcal{D}_i^{\mathcal{E}}, \longrightarrow_{\mathcal{E}})$, where $(\vec{s}_1, \vec{i}_1) \longrightarrow_{\mathcal{E}} (\vec{s}_2, \vec{i}_2)$ iff an ensemble in state $\vec{s}$ and with input $\vec{i}$ from the environment has a transition to state $\vec{s'}$ (i.e., $\exists \vec{o}.\ ((\vec{i}, \vec{s}), (\vec{s'}, \vec{o})) \in \delta_{\mathcal{E}}$) and the environment can generate output $\vec{i'}$ in the next step.*

Definition 21. *Consider n controlled physical environments $E_{M_i} = (C_i, \vec{x}_i, \Lambda_i)$ with $\vec{x}_i = (x_{i_1}, \ldots, x_{i_{l_i}})$ for $i = 1, \ldots, n$. Given a variable t for time, a* time-invariant constraint *is a logic formula of the form $(\forall t)\ \psi(\vec{x}_1(t), \vec{x}_2(t), \ldots, \vec{x}_n(t))$.*