SMT-Based Analysis of Virtually Synchronous Distributed Hybrid Systems

Kyungmin Bae  
SRI International

Peter Csaba Ölveczky  
University of Oslo

Sicun Gao  
MIT CSAIL

Soonho Kong  
Carnegie Mellon University

Edmund M. Clarke  
Carnegie Mellon University

ABSTRACT

This paper presents general techniques for verifying virtually synchronous distributed control systems with interconnected physical environments. Such cyber-physical systems (CPSs) are notoriously hard to verify, due to their combination of nontrivial continuous dynamics, network delays, imprecise local clocks, asynchronous communication, etc. To simplify their analysis, we first extend the PALS methodology—that allows to abstract from the timing of events, asynchronous communication, network delays, and imprecise clocks, as long as the infrastructure guarantees bounds on the network delays and clock skews—from real-time to hybrid systems. We prove a bisimulation equivalence between Hybrid PALS synchronous and asynchronous models. We then show how various verification problems for synchronous Hybrid PALS models can be reduced to SMT solving over nonlinear theories of the real numbers. We illustrate the Hybrid PALS modeling and verification methodology on a number of CPSs, including a control system for turning an airplane.

Keywords

Distributed hybrid systems; SMT; synchronizers; PALS

1. INTRODUCTION

Virtually synchronous distributed hybrid systems consist of a number of distributed controllers—where each controller may interact with its physical environment having continuous dynamics which, furthermore, can be correlated—that should logically behave in a synchronous way. This class of cyber-physical systems (CPSs) includes avionics, robotics, automotive, and medical systems. Designing and analyzing such systems is difficult because of the interrelated continuous behaviors combined with clock skews, network delays, execution times, and so on.

A key step towards achieving manageable modeling and verification techniques for this complex class of CPSs is to extend the PALS framework to distributed hybrid systems. The PALS (physically asynchronous, logically synchronous) methodology [1, 3, 15] was developed to reduce the design and analysis of a virtually synchronous distributed real-time system (i.e., one without continuous behaviors) to the much simpler tasks of designing and analyzing the underlying synchronous models, provided that the network infrastructure can guarantee bounds on computation times, network delays, and imprecision of the local clocks. In this paper we introduce Hybrid PALS for virtually synchronous distributed hybrid systems. Hybrid PALS extends the approach in [5] to achieve a bisimulation equivalence between Hybrid PALS synchronous models and their asynchronous counterparts.

This means that verifying a virtually synchronous distributed hybrid system reduces to verifying its underlying synchronous model. Hybrid PALS allows us to abstract from asynchronous communication, network delays, message buffering, etc. However, the times at which physical states are sampled or actuator commands are sent to the environment cannot be abstracted away. Since these events are triggered by imprecise local clocks, we must also take into account those clocks. Furthermore, the physical environments of different components are often tightly coupled, so that the continuous dynamics of the entire system becomes nonlinear. For these reasons, analysis techniques for event-based systems (such as hybrid automata) or linear systems cannot be easily used to analyze Hybrid PALS models.

This paper presents SMT solving techniques to address the challenges of analyzing Hybrid PALS models. The verification of a synchronous Hybrid PALS model, which involves (nonlinear) ordinary differential equations (ODEs) and clock skews, is reduced to checking the satisfiability of SMT formulas over the real numbers, which is decidable up to any user-given precision [8, 10]. We show how standard verification problems for hybrid systems, such as bounded reachability, unbounded time inductive reasoning, and compositional assume-guarantee reasoning, can be encoded as SMT formulas for synchronous Hybrid PALS models.

We have applied our techniques on a range of non-trivial nonlinear systems, including: a control system for turning an airplane, a networked controller for physically connected water tanks, and a networked thermostat controller for interconnected adjacent rooms. These case studies involve nonlinear ODEs and continuous connections between different components, and take into account network delays, clock skews, asynchronous communication, execution times, etc.
To summarize, the new contributions in this paper (also compared to [5]) are: (i) more refined and complete Hybrid PALS models; (ii) a bisimulation result between synchronous and asynchronous Hybrid PALS models; (iii) general SMT techniques for analyzing synchronous Hybrid PALS models (as opposed to showing only concrete analysis of toy examples in [5]); and (iv) illustrating the effectiveness of Hybrid PALS and the proposed verification methodology on complex examples and hybrid systems benchmarks.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 gives a background on PALS. Section 4 introduces Hybrid PALS. Section 5 shows discussions related work. Section 3 gives a background on PALS and the proposed verification methodology on complex examples and hybrid systems benchmarks. Finally, Section 7 gives some concluding remarks.

2. RELATED WORK

PALS [1, 3, 15] targets distributed real-time systems, whose absence of continuous behaviors means that the timing of events, and hence local clocks, can be abstracted away in the synchronous models, which can therefore be verified by standard model checking techniques. In contrast, (synchronous) Hybrid PALS models must take both continuous behaviors and clock skews into account and therefore cannot be analyzed using such techniques for discrete systems.

The initial steps towards a hybrid extension of PALS were taken in [5]. However, the formal models of Hybrid PALS in [5] are very different from the models in this work, so that a bisimulation equivalence could not be provided in [5]. In this paper, Hybrid PALS models are significantly refined to obtain a bisimulation between synchronous and distributed hybrid models, and to allow more general sampling and response times of sensors and actuators. For example, components in the same synchronous state may have different local times in [5], but those times are synchronized in this paper to properly model the continuous behavior for tightly coupled environments. Furthermore, [5] shows that two interconnected thermostats can be verified using dReal (e.g., the airplane example in our paper), besides symmetric distributed hybrid systems.

Our case studies on networks of identical hybrid systems are related to symmetry-reduction approaches for networks of timed or hybrid automata (e.g., [6, 11, 13]), and their compositional analysis for any number of identical processes is related to [12]. Such work uses hybrid or timed automata where communication is specified using joint synchronous actions, whereas our work focuses on time-triggered systems with nonlinear dynamics where communication is governed by real-time constraints, taking into account network delays, execution times, and clock skews, and where the local environments of tightly coupled components continuously interact with each other. In addition, Hybrid PALS considers general virtually synchronous distributed hybrid systems (e.g., the airplane example in our paper), besides symmetric distributed hybrid systems.

3. PRELIMINARIES ON PALS

PALS transforms a multirate synchronous design SD into a distributed real-time system MA(\(SD, \Gamma\)) that satisfies the same temporal logic properties, provided that the underlying infrastructure assures bounds \(F = (\epsilon, o_{\text{min}}, o_{\text{max}}, p_{\text{min}}, p_{\text{max}})\) with: (i) \(\epsilon \geq 0\) the maximal skew of any local clock with respect to the global clock, (ii) \([o_{\text{min}}, o_{\text{max}}]\) time bounds for executing a transition, and (iii) \([p_{\text{min}}, p_{\text{max}}]\) time bounds for the network transmission delay. This section overviews the synchronous models SD, the distributed models MA(\(SD, \Gamma\)), and their relationship (we refer to [3, 15] for details).

3.1 Discrete Synchronous Models

The synchronous model SD is specified as an ensemble (of nondeterministic) 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) \(D_o = D_{o_1} \times \cdots \times D_{o_n}\) an output set, and (iv) \(\delta_M \subseteq (D_i \times S) \times (S \times D_o)\) a total transition relation.

A collection \(\{M_j\}_{j \in J_1, J_2}\) of state machines with different rates can be composed into a multirate ensemble \(\cal{E}_T\) with global period \(T\), as illustrated in Fig. 1, where the period of a slow typed machine \(s \in J_s\) (with rate(s) = 1) is \(k\) times the period of a fast machine \(f \in J_f\) with rate\((f) = k > 1\).

A wiring diagram connects the input and output ports, so that there are no connections between two fast machines. In each round of \(\cal{E}_T\), all components perform a transition in lockstep. A fast machine \(f\) is slowed down and performs \(k = \text{rate}(f)\) internal transitions in one global synchronous step. Since a fast machine produces \(k\)-tuples of outputs in one step, input adaptors 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 \(k\)-tupple of inputs for a fast machine.

This synchronous composition of an ensemble \(\cal{E}_T\) is thus equivalent to one machine \(M_{\cal{E}_T} = (D_{\cal{E}_T}, S_{\cal{E}_T}, D_{\cal{E}_T}^o, \delta_{M_{\cal{E}_T}})\). If a machine in \(\cal{E}_T\) has a feedback wire connected to itself or to another component, then the output becomes an input of the destination in the next iteration. That is, \(M_{\cal{E}_T}\)'s states \(S_{\cal{E}_T}\) consist of the states of its subcomponents \(M_j\) and the “feedback” outputs. For example, \(M_{\cal{E}_T}\) of the ensemble \(\cal{E}_T\) in Fig. 1 is the machine given by the outer box.

Definition 2. The transition system for \(M_{\cal{E}_T}\) is a tuple \(ts(M_{\cal{E}_T}) = (S_{\cal{E}_T} \times D_{\cal{E}_T}^o, \rightarrow_{\cal{E}_T})\), where \((\vec{s}_1, \vec{s}_2, \vec{r}_1, \vec{r}_2)\) is an element in state \(\vec{s}_1\) with input \(\vec{r}_1\) from the interface has a transition to state \(\vec{s}_2\) (i.e., \(3\delta((\vec{s}_1, \vec{s}_2), (\vec{r}_1, \vec{r}_2)) \in \delta_{M_{\cal{E}_T}})\).

3.2 PALS Distributed Real-Time Models

Each component in the distributed model \(MA(\cal{E}_T, \Gamma)\) is composed of a machine in \(\cal{E}_T\) and wrappers around it, as depicted in Fig. 2. In \(MA(\cal{E}_T, \Gamma)\), each machine performs transitions at its own rate according to its local clock. At the beginning of its period, it reads input from the layer above, performs a transition, and then generates output. A wrapper has I/O buffers, timers, and access to the local clock of the machine. A PALS wrapper has the same period...
T and stores received inputs in its input buffer. When its i-th round begins (at some time in \( (iT - \epsilon, iT + \epsilon) \)), it delivers the contents of its input buffer to its input adaptor wrapper, and sets its backoff timer to \( 2\epsilon - \mu_{\text{min}} \). When the execution of the inner components ends and the backoff timer expires, the contents of the output buffer are sent out.

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

Notice that a fast machine \( M_f \) may not be able to finish all of its \( k \) transitions in a global round, but only \( k' \) transitions before the outputs must be sent. If \( k' < k \), then the \( k \)-machine wrapper only sends the first \( k' \) values. The input adaptor of each input port whose source is \( M_f \) must be \((k' + 1)\)-oblivious: that is, it ignores the last \( k - k' \) values \( v_{k'+1}, \ldots, v_k \) in a \( k \)-tuple \((v_1, \ldots, v_k)\).

**Stable states of \( \mathcal{MA}(E, \Gamma) \)** are snapshots of the system at times \( iT - \epsilon \), just before the components in \( \mathcal{MA}(E, \Gamma) \) start performing local machine transitions [15]. In \( \mathcal{MA}(E, \Gamma) \), network transmission can happen only in the interval \( (iT + \epsilon, (i + 1)T - \epsilon) \). In stable states at times \( iT - \epsilon \), the input buffers of the PALS wrappers are empty, and the other input and output buffers are empty. The function \( \text{sync} \) maps stable states to the corresponding states of \( M_{E_f} \).

**Definition 3.** For a stable state \( C \) of \( \mathcal{MA}(E, \Gamma) \), \( \text{sync}(C) \) is a pair \( \{(s_j, f_{ij})\}_{i,j \in I, j \in F} \in \mathcal{S}^{E_f} \times \mathcal{D}^{E_f} \) such that: (i) the machine states in \( C \) give the states \( \{s_j\}_{i,j \in I, j \in F} \in \mathcal{M}_{E_f} \); and (ii) the values in the input buffers of the PALS wrappers in \( C \) give the values \( \{f_{ij}\}_{i,j \in I, j \in F} \) in the feedback wires and the input \( \vec{f} \) from the ensemble interface in \( \mathcal{M}_{E_f} \).

**Big-step transitions** \( \rightarrow_{M_{\text{HS}}} \) are defined between two stable states, and they are related to single synchronous steps of \( M_{E_f} \). Because of \((k' + 1)\)-obliviousness of the input adaptors, two stable states are related by \( C_1 \sim_{\text{obs}} C_2 \) iff their machine states are identical and their associated input buffer contents cannot be distinguished by input adaptors.

**Theorem 1.** [3] The binary relation \( \sim_{\text{obs}} \) is a bisimulation between the transition system \( ts(M_{E_f}) \) and the big-step transition system \( \text{Stable}(\mathcal{MA}(E, \Gamma), \rightarrow_{M_{\text{HS}}}) \).

## 4. HYBRID PALS

This section introduces Hybrid PALS, which extends PALS to distributed hybrid systems. In PALS, the time when an event takes place does not matter, as long as it happens within a certain time interval. However, in hybrid systems, we cannot abstract from the time when a continuous value is read or an actuator command is given (both of which depend on a component’s local clock), and thus those times are also included in the synchronous Hybrid PALS models.

In Hybrid PALS, the standard PALS models \( \mathcal{MA}(E, \Gamma) \) and \( E_E \) are nondeterministic models defined for all possible environment behaviors. For a physical environment \( E \), the environment restrictions \( \mathcal{MA}(E, \Gamma) \upharpoonright E \) and \( E_E \upharpoonright E \) define the behavior of the models constrained by \( E \). Section 4.5 gives a bisimulation equivalence between \( \mathcal{MA}(E, \Gamma) \upharpoonright E \) and \( E_E \upharpoonright E \) (we refer to the longer report [4] for more details on the definitions and the proof).

### 4.1 Controlled Physical Environments

A state of a physical environment of machine \( M \) is given by a tuple \( \vec{v} = (v_1, \ldots, v_l) \in \mathbb{R}^l \) of its physical parameters \( \vec{x} = (x_1, \ldots, x_l) \). The behavior of \( \vec{x} \) can be modeled by ODEs that specify trajectories \( t_j, \ldots, t_i \) of \( \vec{x} \) over time. A trajectory of duration \( T \) is a function \( \tau : [0, T] \rightarrow \mathbb{R}^l \).

**Definition 4.** A controlled physical environment is a tuple \( E_M = (C, \vec{F}, \Lambda) \), where: (i) \( C \) is a set of control commands; (ii) \( \vec{x} = (x_1, \ldots, x_l) \) is a vector of real number variables; and (iii) \( \Lambda \subseteq (\mathbb{C} \times \mathbb{R}^d \times \mathbb{R}^d) \times \mathbb{C}^d \) is a physical transition relation such that \((a, \vec{v}, t, \vec{f}) \in \Lambda \) iff for a control command \( a \in C \) that lasts for duration \( t \), \( E_M \)'s physical state \( \vec{x} \) follows the trajectory \( \vec{f} \in \mathbb{T}^l \) from state \( \vec{f}(0) = \vec{v} \).

Many physical environments can be physically correlated, and one local environment may immediately affect another environment. Such correlations are naturally expressed as time-invariant constraints \( \forall t. \psi \) of physical parameters over time \( t \). For example, if parameter \( x_1 \) of \( E_M \) must be equal to parameter \( x_2 \) of \( E_M' \), then the time-invariant constraint is the formula \( \forall t. x_1(t) = x_2(t) \) with \( t \) a variable over time.

**Example 1.** Consider a distributed CPS controller to roll an airplane by moving its ailerons (flaps attached to the end of the left or the right wing), illustrated in Fig. 4. Each aileron subcontroller moves the corresponding aileron towards the goal angle given by the main controller. The subcontrollers and the main controller operate at different rates. The objective of the main controller is to roll the aircraft toward the goal angle specified by the pilot.

The physical environment \( E_M \) of each subcontroller \( M \) specifies the dynamics of the aileron angle \( \theta_M(t) \) according to the moving rate \( r_M \) (the control command from \( M \)) by the
ODE $\dot{x}_M = r_M$. The controlled physical environment is given by $E_M = (\mathbb{R}, x_M, \Lambda_M)$, with $\mathbb{R}$ the domain of $r_M$, and $\Lambda_M \subseteq (\mathbb{R} \times \mathbb{R} \times \mathbb{R}_{\geq 0}) \times T$ such that $((t_M, v_M, 15), \tau) \in \Lambda_M$ iff $\forall t \in [0, 15]$. $\tau(t) = t_M + \int_0^t \dot{r}_M \, dt$.

The physical environment $E_{\text{Main}}$ of the main controller specifies the dynamics of the roll angle $\phi$ using the ODEs $\phi = p$ and $\dot{p} = c(\phi - \phi_L)$, disregarding the yawing effect caused by the rolling, where $p$ is the rolling moment, and $\phi_L$ and $\phi_R$ are the angles of respective ailerons. The controlled physical environment is $E_{\text{Main}} = \{\ast\}$, indicating that $E_{\text{Main}}$ has no control command, and $\Lambda_{\text{Main}} \subseteq \{\ast\}$, with the singleton $\{\ast\}$, and $\Lambda_{\text{Main}}$ is a collection of ODEs and $\phi/\dot{\phi}$ such that $((\ast, (v_M, \phi_M, \phi_L, \phi_R)), 60, (\dot{v}_M, \dot{\phi}_M, \dot{\phi}_L, \dot{\phi}_R)) \in \Lambda_{\text{Main}}$.

$\forall t \in [0, 60]$, $\tau_p(t) = \frac{v_0 - v_p}{\tau_p} + \int_0^t \frac{\tau_p(t)}{c(\tau_R(t) - \tau_L(t))} \, dt$.

The control angles $\phi_L$ and $\phi_R$ must always be the same as the respective aileron angles $x_I$ and $x_R$ of the subcontrollers. However, since the main controller and the subcontrollers have different periods with local clock skew, the ODEs of the subcontrollers cannot be “plugged” into $E_{\text{Main}}$. Instead, their physical correlations are specified by the time-invariant constraint: $\forall t, (\phi_L(t) = x_I(t)) \land (\phi_R(t) = x_R(t))$.

4.2 Sampling and Response Timing

A controller $M$ interacts with its physical environment $E_M$ according to its local clock, which may differ from the global time by up to the maximal clock skew $\epsilon$. Let $c_M : \mathbb{N} \rightarrow \mathbb{R}_{>0}$ denote the global time $c_M(i)$ at the beginning of the $(i + 1)$-th period according to $M$’s local clock.

Fig. 5 depicts the behavior of $M$ with respect to $E_M$ in a PALS distributed model for an interval $[iT - \epsilon, (i + 1)T - \epsilon]$ for its period $T$. The $(i + 1)$-th period of $M$ begins at time $c_M(i) \in (iT - \epsilon, iT + \epsilon)$. Because of PALS bounds, $M$ has already received all the inputs $\vec{i}$ before time $iT - \epsilon$. Next, the physical state $\vec{v}_M$ of $E_M$ is read at time $c_M(i) + t_I$ for some sampling time $t_I$. $M$ then executes its transition based on the inputs $\vec{i}$, the sampled state $\vec{v}_M$, and the machine state $s$.

After the execution, the machine state changes to $s'$, and the new controller command $a$ is sent to $E_M$ at time $c_M(i) + t_R$ for some response time $t_R$ for the execution and the actuator processing. The new outputs $\vec{a}$ from $M$ are delivered to their destinations for the next round before time $(i + 1)T - \epsilon$.

We assume that a control command from $M$ to $E_M$ only depends on $M$’s current state $s$. In Fig. 5, a current control command $a$ (by state $s$) remains effective until the execution of $M$ ends at time $u_R = c_M(i) + t_R$, and then a new control command $a'$ (by state $s'$) takes effect. That is, $E_M$ defines trajectories $T$ of its physical parameters $\vec{x}$ in a time interval $[iT - \epsilon, (i + 1)T - \epsilon]$ of duration $T$ with respect to $(a, a', u_R)$.

Definition 5. Trajectories $T$ of duration $T$ are realizable with respect to commands $a, a' \in C$ and a response duration $u \in \mathbb{R}$ for $E_M$, denoted by $T \in \mathcal{R}_T^{E_M}(a, a', u)$, if $(\vec{a}, \vec{a}(0), u, \vec{a} \subseteq u) \in \Lambda$ and $((a', \vec{a}(u), T - u), \vec{a} \supseteq u) \in \Lambda$.

4.3 Environment-Restricted Controllers

A controller $M$ is basically a nondeterministic machine parameterized by any behavior of its environment $E_M$. A controller $M$ has a state space of the form $\mathbb{S} \times \mathbb{R}_m$, where $\mathbb{R}_m$ denotes the physical parameters that $M$ can observe. Likewise, $E_M$ has a state space of the form $\mathbb{R} = \mathbb{R} \times \mathbb{R}_{\geq 0}$, where $\mathbb{R}$ is the observable part of $\mathbb{R}$. For state $\vec{v} \in \mathbb{R}$ of $E_M$, let $\pi(\vec{v}) \in \mathbb{R}$ denote the observable part of $\vec{v}$.

The environment restriction $M | E_M$ is a normal machine (see Def. 1) with a combined state space $\mathbb{S} \times \mathbb{R}$, recording “snapshots” of $E_M$’s states at times $iT - \epsilon$. A transition of $M | E_M$ from state $(s, \vec{v})$ to $(s', \vec{v}')$ corresponds to realizable trajectories $T'$ for its period $T$ with respect to $\pi_1(s)$ and $\pi_2(s')$ and the response duration $u_R = (c_M(i) + r_M(s) - (iT - \epsilon)$.

The controller $M$ performs a transition based on the observable physical state $\pi_1(T'(s))$ of $E_M$ sampled after the sampling duration $u_I = (c_M(i) + r_M(s) - (iT - \epsilon)$.

Definition 7. The environment restriction of $M$ by $E_M$ with an interface $\pi$ is $M |_E E_M = (D_1, S \times \mathbb{R}_m, D_0, \delta_{M \cup E_M})$, where $((i, (s, \vec{v})), ((s', \vec{v}'), \vec{a})) \in \delta_{M \cup E_M}$ if for the round numbers $i \geq \pi_2(s)$ and $\pi_2(s') = i + 1$, the sampling duration $u_I = (c_M(i) + r_M(s) - (iT - \epsilon)$, and the response duration $u_R = (c_M(i) + r_M(s) - (iT - \epsilon)$:

(i) $T(0) = \vec{v}$ and $T(T) = \vec{v}'$ for some realizable trajectories $T \in \mathcal{R}_{E_M}^T(\pi_1(s), u_R, \pi_2(s'))$, and (ii) $((\vec{v}', (s, \pi_1(T'(s))))((s', \pi_2(T'(s'))), (\vec{a}, \vec{a}(\vec{v}')), \vec{a})) \in \delta_M$.
The interface $\pi_M$ is defined by the projection functions:
(i) $\pi_{\mathcal{C}}(i, r_M, v_M) = r_M$; (ii) $\pi_{\mathcal{T}}(i, r_M, v_M) = i$ (the round number); (iii) $\pi_I(i, r_M, v_M) = 1$ ms (the sampling time); and (iv) $\pi_R(i, r_M, v_M), g_M) = 2$ ms (the response time). Thus, the sampling duration is $u_I = (c_M(i) + 1) - (i \cdot 15 - \epsilon)$ and the response duration is $u_R = (c_M(i) + 2) - (i \cdot 15 - \epsilon)$ for each round. For $E_M = (\mathbb{R}, x_M, \Lambda)$ in Example 1, its observable state is given by the function $\pi_0(x_M) = x_M$.

The environment restriction of $M$ by $E_M$ is then defined as the machine $M|_{\pi_M} E_M = (\mathbb{R}, N \times \mathbb{R}^3, R, \delta|_{\pi_M}(x_M), (g_M, (i, r_M, v_M)), (i', r_M', v_M'), (v_M', v_M), (v_M, v_M')) \in \delta_M|_{\pi_M}(x_M)$ if:
(i) for some realizable trajectory $r \in R_{\delta|_{\pi_M}(x_M)}$ (r_M, r_M, u_M), $v_M = \tau(0)$ and $v_M' = \tau(15)$; and (ii) an $M'$ transition

Example 3. Consider the main controller $M_{Main}$ to roll the aircraft toward the specified angle in Example 1. In each 60 ms round, beginning at $c_{Main}(i) \in (i \cdot 60 \text{ ms} - e, i \cdot 60 \text{ ms} + e)$, $M_{Main}$ receives a desired roll angle $\varphi$ and the aileron angles $(v_L, v_R)$, and sends the new goal angles $(g_L, g_R)$. $M_{Main}$ is specified as the typed machine $(\mathbb{R}^3, N \times \mathbb{R}^3, R^3, \delta_{Main})$, where
\[ (\{\varphi, v_L, v_R\}, (i, \varphi)) \big( (i', \varphi'), (v_L, g_L, g_R) \big) \big) \in \delta_{Main} \text{ holds iff:} \]
(i) $\pi_{\mathcal{C}}(i, \varphi) = *$ (no control commands); (ii) $\pi_{\mathcal{T}}(i, \varphi) = i$; and (iii) $\pi_I(i, \varphi) = 3$ ms; and (iv) $\pi_R(i, \varphi), \delta_{Main}$ holds iff:
$g_R = 0.3 g_L - v_L$, $g_L = \ln - g_R$, $\sigma = v_L$, and $i + 1$.

The interface $\pi_{\mathcal{E}}$ is defined by the projection functions:
(i) $\pi_{\mathcal{C}}(i, \varphi) = *$; (ii) $\pi_{\mathcal{T}}(i, \varphi) = i$; and (iii) $\pi_I(i, \varphi) = 3$ ms; and (iv) $\pi_R(i, \varphi), \Delta_\mathcal{E}$ holds iff:
$g_R = 0.3 g_L - v_L$, $g_L = \ln - g_R$, $\sigma = v_L$, and $i + 1$.

4.4 Hybrid PALS Synchronous Models

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

Definition 8. A hybrid multirate ensemble $E\mathcal{T}$ on $E$ is composed of:
(i) $E\mathcal{T}$ a multirate ensemble, (ii) a family of interface functions $\Pi = \{(\pi_i)_{i \in J} \cup J, \lambda\}$, and (iii) a family of physical interfaces $E = \{(E_M)_{i \in J} \cup J, \lambda\}$ with $(\forall \psi)$ the time-invariant constraints (shown in Fig. 6).

Example 4. For the simple CPS controller in Example 1, $E_{Main}$ is a multirate ensemble of the main controller $M_{Main}$ and the subcontrollers $M_L$ and $M_R$, where $rate(Main) = 1$ and $rate(L) = rate(R) = 4$. The machines $\{M_{Main}, M_L, M_R\}$, interfaces $\Pi = \{\pi_{Main}, \pi_L, \pi_R\}$, and physical environments $\{E_{Main}, E_L, E_R\}$ are defined in Examples 2–3. The network connections are shown in Fig. 4. The time-invariant constraints $(\forall \psi)$ $\lambda(L(t)) = \lambda(L(t))$ and $\lambda(R(t)) = \lambda(R(t))$ specify the continuous physical connections.

A hybrid ensemble $E\mathcal{T}$ on $E$ induces a normal multirate ensemble $E\mathcal{T}|_{E\mathcal{T}}$ on $E$, which disregards the time-invariant constraints $(\forall \psi)$. The behaviors of $E\mathcal{T}|_{E\mathcal{T}}$ on $E$ are a subset of the behaviors of $E\mathcal{T}|_{E\mathcal{T}}$ on $E$, namely, the behaviors restricted by $(\forall \psi)$. As mentioned, a transition of a (decelerated) environment-restricted machine $M_i|_{x_E} E_M$ corresponds to realizable trajectories $r$ in a time interval $[iT - \epsilon, (i + 1)T - \epsilon]$ for a global period $T$. Hence, a lockstep composition of such transitions whose realizable trajectories $r$ also satisfy the time-invariant constraints $(\forall \psi)$ gives a transition of the synchronous composition $M_{sys}|_{x_E}$.

4.5 Hybrid PALS Distributed Models

Hybrid PALS maps a hybrid ensemble $E\mathcal{T}$ on $E$ to the distributed hybrid system $MA(E\mathcal{T}, \Gamma)|_{x_E}$, together with PALS bounds $\Gamma$; i.e., $(E\mathcal{T}|_{x_E}) \rightarrow MA(E\mathcal{T}, \Gamma)|_{x_E}$.

The distributed components in $MA(E\mathcal{T}, \Gamma)|_{x_E}$ are exactly the same as the wrapper hierarchies of Fig. 2 in $MA(E\mathcal{T}, \Gamma)$.

But the controllers in $MA(E\mathcal{T}, \Gamma)|_{x_E}$ also interact with their physical environments in $E$, according to the sampling and response timing policy $\Pi$ to determine sensor sampling timing $t_i$ and actuator response timing $t_R$.

The behaviors of the Hybrid PALS distributed hybrid $MA(E\mathcal{T}, \Gamma)|_{x_E}$ are the subset of those of the PALS distributed system $MA(E\mathcal{T}, \Gamma)$, restricted by the physical environments and the time-invariant constraints in $E$. The continuous dynamics of $MA(E\mathcal{T}, \Gamma)|_{x_E}$ is “completely” decided by the controllers, their physical environments, the timing policies, and the local clocks of the controllers.

More precisely, consider a normal ensemble $E\mathcal{T}|_{E\mathcal{T}}$ above, induced by a hybrid ensemble $E\mathcal{T}|_{E\mathcal{T}}$, and composed of the environment-restricted controllers $\{M_i|_{x_E} E_M\}_{i \in J} \cup J$. A big-step transition is defined from one stable state at time $iT - \epsilon$ to another stable state at $(i + 1)T - \epsilon$ for the system $MA(E\mathcal{T}, \Gamma)|_{x_E}$, as explained in Section 3.2. For such time intervals $[iT - \epsilon, (i + 1)T - \epsilon]$, each $M_i|_{x_E} E_M$ provides (k-step) realizable trajectories $\vec{r}_i$, based on round numbers, control commands, sampling timings, and response timings. The behaviors of $MA(E\mathcal{T}, \Gamma)|_{x_E}$ are the behaviors of $MA(E\mathcal{T}, \Gamma)|_{x_E}$ that are restricted by the time-invariant constraints $\Gamma$. Big-step transitions of $MA(E\mathcal{T}, \Gamma)|_{x_E}$ are therefore exactly those of $MA(E\mathcal{T}, \Gamma)|_{x_E}$ whose associated trajectories $\vec{r}_i$ satisfy the time-invariant constraints $(\forall \psi)$.

The correctness of Hybrid PALS follows from the fact that each physical measurement and physical activation happens at the same time in both $E\mathcal{T}|_{x_E}$ and $MA(E\mathcal{T}, \Gamma)|_{x_E}$ with the same timing policies $\Pi$ (see [4] for more details).

Theorem 2. The relation $(\sim_{obs}; \text{sync})$ is a bisimulation between the transition system $TS(M_{E\mathcal{T}, \Gamma})$ and the big-step transition system induced by $MA(E\mathcal{T}, \Gamma)|_{x_E}$, exhibiting the exactly same set of realizable trajectories.
state $C$ of $\mathcal{MA}(E,T,\Gamma)$ in $E$. By definition, there exists $s \rightarrow E_T|\in E$ of $\mathcal{MA}(E_T|\in E)$ with some realizable trajectories $\tau$ that satisfy the time-invariant constraints $(\forall t) \psi$ for the induced ensemble $E_T|\in E$. By Theorem 1, $(\tau_{\text{obs}}; \text{synct})$ is a bisimulation between the transition systems $ts(\mathcal{MA}(E_T|\in E))$ and $(\text{Stable}(\mathcal{MA}(E_T|\in E, \Gamma)), \rightarrow)$. Therefore, for some stable state $C'$, there is a big-step transition $C \rightarrow_{C'} C'$ in $\mathcal{MA}(E_T|\in E, \Gamma)$ such that $s \rightarrow E_T|\in E$ and $C \rightarrow_{C'} C'$ involve exactly the same machine states and inputs. By construction, round numbers, control commands, sampling timings, and response timings all depend on machine states and inputs. Therefore, the trajectories $\tau$, which satisfy $(\forall t) \psi$, are also realizable for $C \rightarrow_{C'} C'$, and there exists a big-step transition $C \rightarrow_{C'} C'$ by $\mathcal{MA}(E_T, \Gamma)$ in $E$. The other direction is similar. □

5. HYBRID PALS MODELS IN SMT

Theorem 2 implies that analyzing $\mathcal{MA}(E_T, \Gamma)$ in $E$ can be reduced to the much simpler problem of analyzing $\mathcal{MA}(E_T|\in E)$, which abstracts from asynchronous communication, network delays, execution times, message buffering, etc. This section shows how analysis problems for a hybrid ensemble $E_T$ in $E$ can be encoded as formulas over the real numbers and ODEs. We symbolically encode all possible local clocks to deal with clock skew.

5.1 Encoding Hybrid PALS Models

5.1.1 Encoding Environment-Restricted Controllers

A controller $M$ can be expressed as a formula of the form $\phi_M(\vec{i}, \vec{y}, \vec{y}', \vec{o})$, with variables $\vec{i}$, $\vec{y}$, $\vec{y}'$, and $\vec{o}$ denoting input, output, and, respectively, in such a way that $\phi_M(\vec{i}, \vec{s}, \vec{s}', \vec{o}) \iff (\vec{i}, \vec{s}, (\vec{s}', \vec{o})) \in \delta_M$.

Example 5. For an aileron subcontroller $M$ in Example 2, the formula $\phi_M(\vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M) \equiv (\vec{y}_M = (y_M - y_0)_1/15)$ is given by

\[
\forall t \in [y_0, y_1], \quad y_M(t) = y_0 + \int_0^t \phi_M(\vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}) dt.
\]

If the continuous dynamics of $E_T$ is described by a system of ODEs $\frac{d\vec{y}}{dt} = F_0(\vec{x}, t)$ for a control command $\vec{a}$ and an interval $[a_0, a_1]$, then $\phi_M$ includes universal quantification over time along with solutions of the ODEs, for example:

\[
\forall t \in [a_0, a_1], \quad \phi(t) = a + \int_0^t \phi_M(\vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}) dt.
\]

Example 6. For an aileron subcontroller $M$ in Example 1, the formula $\phi_M(\vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M) \equiv (\vec{y}_M = (y_M - y_0)_1/15)$ is given by:

\[
\forall t \in [y_0, y_1], \quad y_M(t) = y_0 + \int_0^t \phi_M(\vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}) dt.
\]

For the main controller $M$, where $y_0 = 0$, then the formula $\phi_M(\vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M, \vec{y}_M) \equiv (\vec{y}_M = (y_M - y_0)_1/15)$ is:

\[
\forall t \in [0, y_1], \quad y_M(t) = y_0 + \int_0^t \phi_M(\vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}, \vec{y}) dt.
\]

The other direction is similar.

5.2 Encoding Hybrid Ensembles

Recall that all machines in a synchronous composition $E_T$ perform their transitions in lockstep; for a fast machine $k$, $f_k = rate(f)$ transitions in a global round. The encoding $\phi_{E_T}^{\mathcal{MA}(E_T, \Gamma)}$ of the k-step deceleration of an environment restriction $E_T \models E_M$ is given by sequentially composing the $k$ formulas $\phi_{E_T}^{\mathcal{MA}(E_T, \Gamma), k}$ for the subintervals $[ik + n - 1, ik + n]$ for $n = 1, \ldots, k$.
A hybrid ensemble $\mathcal{E}_T$ \(\in E\) of typed machines \(\{M_f\}_{j \in J}\), physical environments \(\{E_L\}_{i \in I}\), for \(J = J_S \cup J_F\), and the time-invariant constraint \((\forall t \cdot \psi)\) is encoded as a formula $\phi_{\mathcal{E}_T | \mathcal{E}_L} \in E_i$ as follows:

\[
\exists (i, \vec{o}_i) \in J \cap I \exists (\vec{y}_i, v_i^o, f_i, \vec{f}_i) \in J \cup I \exists (\vec{y}_j, v_j^{f_i}, \vec{f}_j) \in J \cup I \exists (\vec{y}_j, v_j^f) \in J \cup I \exists (\vec{y}_k, v_k^f) \in J \cup I \forall (t_i) \exists (\vec{y}_i, v_i^o, f_i, \vec{f}_i) \in J \cup I \exists (\vec{y}_j, v_j^{f_i}, \vec{f}_j) \in J \cup I \exists (\vec{y}_j, v_j^f) \in J \cup I \exists (\vec{y}_k, v_k^f) \in J \cup I \forall (t_i)
\]

\[
\phi_{\mathcal{E}_T | \mathcal{E}_L} = \phi_{M_{T,F}^{\rightarrow}}(\vec{y}_i, v_i^o, f_i, \vec{f}_i) \quad \wedge \quad \wedge \phi_{\text{ver}(\vec{t}, \vec{d}, \vec{o}_i, \vec{f}_i, \vec{f}_j)} \wedge \wedge (\forall t_i)
\]

Theorem 3. The formula $\phi_{\mathcal{E}_T | \mathcal{E}_L}$ is satisfiable if there exists a corresponding transformation of the synchronous composition $M_{T,F}^{\rightarrow}$ from $T = \vec{t}$ to $\vec{t} + \epsilon$ for some local clocks.

Theorem 2.4. The formula $\phi_{\mathcal{E}_T | \mathcal{E}_L}$ is satisfiable if there exists a corresponding stable transformation for some local clocks.

5.2.3 Inductive Reasoning

For unbounded time verification, we encode an inductive proof of a safety property $\text{safe}(\vec{y}, \vec{\tau}(t))$ as a logical formula by using an inductive condition for $M_{T,F}^{\rightarrow}$'s synchronous transitions, which implies the safety property. An inductive invariant consists of two formulas: $\text{ind}_d(\vec{y})$ for state variables $\vec{y}$, and $\forall t \in [0, T], \text{ind}_d(\vec{\tau}(t))$ for trajectories $\vec{\tau}$.

Definition 9. Boundedly reachable property of $M_{T,F}^{\rightarrow}$ up to $n$ rounds for a safety property $\text{safe}(\vec{y}, \vec{\tau}(t))$, an initial condition $\text{init}(\vec{y})$, and an input constraint $\in_i(\vec{y})$ is encoded by:

\[
\exists (\vec{y}_k, \vec{\tau}_k, \vec{o}_k, \vec{\tau}_i) \in I \supseteq (\vec{y}_k \in \vec{y}_k, \vec{\tau}_k, \vec{o}_k, \vec{\tau}_i) \in \text{init}(\vec{y}_k) \wedge \bigwedge_{k=1}^n M_{T,F}^{\rightarrow}(\vec{y}_k, \vec{\tau}_k, \vec{o}_k) \in \text{safe}(\vec{y}_k, \vec{\tau}_k) \wedge N_{k=1}^n \text{init}(\vec{y}_k, \vec{\tau}_k, \vec{o}_k, \vec{\tau}_i)
\]

5.2.4 Compositional Reasoning

We encode a divide-and-conquer proof to verify a safety property using standard assume-guarantee reasoning. It is very useful for dealing with the state explosion problem. In $M_{T,F}^{\rightarrow}$ each component $M_{i,j} \in E_{M_i}$ performs a transition based on its input $i_j$ and trajectory $\vec{\tau}_i$ that are restricted by time-invariant constraints. Hence, we use an input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$ and an output condition $c_{\text{out}}(\vec{\tau}_i)$, which satisfy necessary constraints for compositional reasoning:

(i) assuming an input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$, if a transition is taken, then the output condition $c_{\text{out}}(\vec{\tau}_i)$ holds; and

(ii) the collection of the output conditions $\{c_{\text{out}}(\vec{\tau}_i)\}$ implies each input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$.

5.2.5 Composition Reasoning

We encode a divide-and-conquer proof to verify a safety property using standard assume-guarantee reasoning. It is very useful for dealing with the state explosion problem. In $M_{T,F}^{\rightarrow}$ each component $M_{i,j} \in E_{M_i}$ performs a transition based on its input $i_j$ and trajectory $\vec{\tau}_i$ that are restricted by time-invariant constraints. Hence, we use an input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$ and an output condition $c_{\text{out}}(\vec{\tau}_i)$, which satisfy necessary constraints for compositional reasoning:

(i) assuming an input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$, if a transition is taken, then the output condition $c_{\text{out}}(\vec{\tau}_i)$ holds; and

(ii) the collection of the output conditions $\{c_{\text{out}}(\vec{\tau}_i)\}$ implies each input condition $c_{\text{in}}(i_j, \vec{\tau}_i)$.
Definition 11. For each component \( j \) in a hybrid ensemble \( \tilde{E}_r \upharpoonright [t, \delta) E \), consider a safety property \( \text{safe}_{c_j}(\tilde{y}_j, \tilde{\tau}_j(t)) \), an input constraint \( \text{init}_j(\tilde{y}_j) \) for input from \( \tilde{E}_r \)’s interface, and an initial condition \( \text{init}_j(\tilde{y}_j) \). I/O conditions are given by:

\[
\forall t. \ c_{\text{in}}^j(\tilde{y}_j, \tilde{\tau}_j(t)) \equiv (c_{\text{in},d}(\tilde{y}_j) \land \forall t \in [0, T]. \ c_{\text{in},c}(\tilde{\tau}_j(t)))
\]

\[
\forall t. \ c_{\text{out}}^j(\tilde{\sigma}_j, \tilde{\tau}_j(t)) \equiv (c_{\text{out},d}(\tilde{y}_j^d) \land \forall t \in [0, T]. \ c_{\text{out},c}(\tilde{\tau}_j(t)))
\]

that satisfy the necessary constraints for a compositional reasoning of the synchronous composition \( Me \cup E \):

1. \( \forall y_j, \tilde{\sigma}_j, \tilde{y}_j, (\forall t \in [0, T], c_{\text{in}}^j(\tilde{y}_j, \tilde{\tau}_j(t)) \land \phi_{\text{in}}^j(\tilde{y}_j, \tilde{y}_j, \tilde{\sigma}_j, \tilde{\tau}_j(t)) \implies \forall t \in [0, T], c_{\text{out}}^j(\tilde{\sigma}_j, \tilde{\tau}_j(t))) \)

Both bounded reachability and inductive reasoning can then be separately performed for each component by also assuming input conditions for individual components.\(^3\) A compositional bounded reachability problem up to \( n \in \mathbb{N} \) rounds for component \( j \) is encoded by:

\[
\exists \tilde{y}_j \in \{ \tilde{y}_j^1, \tilde{y}_j^2, \tilde{y}_j^3 \}^{n-1}, \text{init}_j(\tilde{y}_j^1) \land \bigwedge_{k=1}^{n} \neg\text{safe}_c(\tilde{y}_j^1, \tilde{\tau}_j^1(t_k))
\]

\[
\land \bigwedge_{k=1}^{n} \left[ \phi_{\text{in}}^j(\tilde{y}_j^k, \tilde{y}_j^{k+1}, \tilde{\tau}_j^k, \tilde{\tau}_j^{k+1}) \land \text{init}_j(\tilde{y}_j^k) \right] \land \forall t \in [0, T]. \ c_{\text{in}}^j(\tilde{y}_j^1, \tilde{\tau}_j^1(t))
\]

where the input condition \( \forall t \in [0, T], c_{\text{in}}^j(\tilde{y}_j, \tilde{\tau}_j(t)) \) is also assumed for each step in addition to Definition 9.

Likewise, a compositional inductive reasoning problem of \( \text{safe}_c(\tilde{y}_j, \tilde{\tau}_j(t)) \) for component \( j \) with an inductive condition \( \text{ind}_d^j(\tilde{y}_j) \land \forall t \in [0, T], \text{ind}_d^j(\tilde{\tau}_j(t)) \) can be encoded by:

\[
\forall y_j, \text{init}_j(\tilde{y}_j) \implies \text{ind}_d^j(\tilde{y}_j)
\]

\[
\forall y_j, \tilde{y}_j, \tilde{\tau}_j, \tilde{\sigma}_j, \left[ \text{ind}_d^j(\tilde{y}_j) \land \phi_{\text{in}}^j(\tilde{y}_j^1, \tilde{y}_j^2, \tilde{\sigma}_j, \tilde{\tau}_j) \land \text{init}_j(\tilde{y}_j) \land \forall t \in [0, T], c_{\text{in}}^j(\tilde{y}_j, \tilde{\tau}_j(t)) \right] \]

\[
\implies (\text{ind}_d^j(\tilde{y}_j) \land \forall t \in [0, T], \text{ind}_d^j(\tilde{\tau}_j(t)))
\]

\[
\forall y_j, \forall t. \text{ind}_d^j(\tilde{y}_j) \implies \text{ind}_d^j(\tilde{\tau}_j(t)) \implies \text{safe}_c(\tilde{y}_j, \tilde{\tau}_j(t))
\]

The formulas for compositional reasoning can also be proved by checking the unsatisfiability of their negated versions.

5.3 Removing Universal Quantification

The formulas encoding Hybrid PALS models may contain universal quantification over \( \text{uninterpreted functions on the real numbers} \), such as time-invariant constraints \( \forall t. \psi \) or formulas of the form \( \forall t \in [u_0, u_1], \tilde{\tau}(t) = \tilde{v} + \int_0^t f_{\tilde{x}}(\tilde{x}(t)) dt \), which are not directly supported by current state-of-the-art SMT techniques. This section explains how such universal quantification can be removed from the formula in this way.

Now consider universally quantified formulas of the form \( \forall t \in [u_0, u_1], \tilde{\tau}(t) = \tilde{v} + \int_0^t F_{\tilde{x}}(\tilde{x}, t) dt \). Recall that the term \( F_{\tilde{x}}(\tilde{x}, t) \) may include “uninterpreted” functions whose meaning is defined by other components. We assign to a time interval \([u_0, u_1]\) a global round such a “partial” ODE system \( \frac{dx}{dt} = F_{\tilde{x}}'(\tilde{x}, t) \). If every component \( j \) assigns its partial ODE system \( \frac{dx}{dt} = F_{\tilde{x}}(\tilde{x}, t) \) to its interval \([u_0, u_1]\), then a complete ODE system \( \frac{dx}{dt} = F_{\tilde{x}}(\tilde{x}, t) \), which contains no uninterpreted functions, can be constructed for the common interval \([u_0, u_1]\), provided that variables are renamed by the equality time-invariant constraints.

Example 9. The physical environment \( E_{\text{Main}} \) of the main controller in Example 1 involves the formula \( \forall t \in [0, 0.6], \]

\[
\left[ \begin{array}{c} \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \end{array} \right] = \left[ \begin{array}{c} \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \end{array} \right] + \int_0^{u_0-u_R} \left[ \begin{array}{c} \tau_{\phi}(t) \\ \tau_{\phi}(t) \\ \tau_{\phi}(t) \end{array} \right] \frac{\tau_{\phi}(t)}{\tau_{\phi}(t)} dt.
\]

This system indicates a period that the right subcontroller has responded (with the new rate \( \tau_{\phi}' \)) but the left one has not responded yet (due to the clock skews).

More precisely, a global period \([0, T]\) for one round of an ensemble \( \tilde{E}_r \upharpoonright [t, \delta) E \) is divided into \( N \) contiguous subintervals \([0, t_1], [t_1, t_2], [t_2, t_3], \ldots, [t_{N-1}, T] \). Each interval denotes a single time segment to which a complete system of ODEs is assigned. The number \( N \) is determined by the total number of interval assignments in one round, namely, a number of ODE subformulas \( \forall t \in [u_0, u_1], \tilde{\tau}(t) = \tilde{v} + \int_0^t f_{\tilde{x}}(\tilde{x}(t)) dt \) in the formula \( \phi_{\tilde{E}_r \upharpoonright [t, \delta) E} \). Finally, we can syntactically build a complete ODE system for each time segment by enumerating all possible combinations of partial ODE systems.

6. CASE STUDIES

This section gives an overview of some case studies that use our methodology to verify virtually synchronous distributed hybrid systems. All the case studies involve nonlinear ODEs and continuous interactions between distributed components. They also take into account asynchronous communication, network delays, clock skews, execution times, etc. Owing to the bisimulation equivalence, we can analyze the simpler synchronous models \( \tilde{E}_r \upharpoonright [t, \delta) E \) instead of analyzing the distributed hybrid models \( MA(\tilde{E}_r, \Gamma) \upharpoonright [t, \delta) E \).

We have verified safety properties using inductive and compositional SMT encodings for any possible set of local clocks with maximal clock skew \( \epsilon \). We have applied the dReal SMT solver [9] to check the satisfiability of the SMT formulas up to a given precision \( \delta > 0 \) (which is decidable for nonlinear hybrid systems [8, 10]). All experiments were conducted on an Intel Xeon 2.0 GHz with 64 GB memory. The case studies and the experimental results are available at http://dreal.github.io/benchmarks/networks.
6.1 Turning an Airplane

We consider a multirate virtually synchronous distributed controller to turn an airplane (adapted from [2]). This is a more elaborate version of Example 1. 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 flap 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, as illustrated in Fig. 7. The desired safety property is that the yaw angle $\beta$ is always close to 0.

Each subcontroller $M$ gradually moves itself towards the goal angle $\text{goal}_L$ specified by the main controller $M_{\text{Main}}$, as explained in Example 1. In each round, $M$ receives $\text{goal}_M$ from $M_{\text{Main}}$, determines the moving rate $\text{rate}_M$ based on $\text{goal}_L$ and the current sampled value $v_M$ of the angle $\alpha_M$, and sends back $v_M$ to $M_{\text{Main}}$. The local environment $E_M$ specifies the dynamics of $\alpha_M$ by the ODE $\dot{\alpha}_M = \text{rate}_M$. The main controller $M_{\text{Main}}$ determines the goal angles for the subcontrollers to make a coordinated turn. In each round, $M_{\text{Main}}$ receives a desired direction $\text{goal}_\phi$ (from the pilot) and the angles ($v_L, v_R, r$) from the subcontrollers, and sends back the new goals ($\text{goal}_L, \text{goal}_\psi, \text{goal}_R$), based on the current sampled position values ($v_L, v_R, \psi$) of the direction angle $\psi$, the roll angle $\phi$, and the yaw angle $\beta$. We use a simple control logic to decide the new goal angles based on the current position angles, namely, by using some function ($\text{goal}_L, \text{goal}_\psi, \text{goal}_R = f_{\text{Main}}(v_L, v_R, \psi)$).

The environment $E_{\text{Main}}$ specifies the lateral dynamics of an aircraft as the nonlinear ODEs (dynamics in Fig. 8).

$$
\dot{\beta} = Y_{\text{Left}} \cdot \dot{\psi} - \frac{r}{m} \sin \phi,
$$

$$
\dot{\psi} = \frac{g}{V} \tan \phi,
$$

$$
\dot{p} = (c_1 r + c_2 p) \cdot \tan \phi + c_3 L_{\text{Left}} \cdot \dot{\psi} \cdot \dot{\beta},
$$

$$
\dot{r} = (c_3 p - c_4 r) \cdot \tan \phi + c_4 L_{\text{Left}} \cdot \dot{\psi} \cdot \dot{\beta},
$$

where $p$ is the rolling moment, $r$ is the yawing moment, and $L_{\text{Left}}, L_{\text{Right}}$, $\dot{\psi}$, and $\dot{\beta}$ are (linear) functions of the control angles ($\psi$, $\phi$, $\beta$).

The physical environment $E_{\text{Main}}$ clearly applies the subcontrollers’ physical environments. Each control angle $\zeta_M$ in $E_{\text{Main}}$ must be the same as the corresponding surface angle $\alpha_M$, but the ODEs of the subcontrollers cannot be directly “plugged” into $E_{\text{Main}}$, because the main controller and the subcontrollers have different periods with local clock skew.

The continuous connections between the physical environments are specified by the time-invariant constraint: $\forall t, (\xi(t) = \alpha_L(t) \wedge \zeta(t) = \alpha_R(t) \wedge (\zeta_R(t) = \alpha_R(t))$.

6.2 Networked Water Tank Controllers

We first performed bounded reachability analysis to verify the safety property $\forall t, \text{abs}(\beta(t)) < 0.2$, where all state variables are initially 0$^\circ$ and the goal direction from the pilot is fixed (e.g., $30^\circ$). In the analysis, we assume the sampling time $t_L = 0$ ms for every controller, the response time $t_R = 3$ ms for every subcontroller (the main controller has no actuator), and the maximal clock skew $\epsilon = 0.2$ ms. For bound $k = 10$, the analysis took 16 hours using dReal with precision $\delta = 0.0001$, which is quite slow due to complex nonlinear ODEs, nontrivial discrete controls, etc.

Therefore, we have applied compositional reasoning to conduct a bounded reachability analysis in a compositional way. We first show that each subcontroller cannot abruptly change its surface angle towards its goal direction in one round, so that the change is always less than a certain value.5 Next, assuming that a subcontroller cannot abruptly move its surface, we perform a bounded reachability analysis only for the main controller using the same initial condition. For bound $k = 20$, using dReal with precision $\delta = 0.0001$, the compositional bounded reachability analysis for the safety property $\forall t, \text{abs}(\beta(t)) < 0.2$ took 2 minutes.
We have proved this compositional safety property for maximal clock skew $\epsilon = 30$ ms, sampling time $t_s = 20$ ms, and response time $t_r = 100$ ms, with precision $\delta = 0.001$ using dReal (the analysis took 4.3 seconds). However, if $\epsilon = 150$ ms, then the inductive condition $(\forall t) x_i(t) \in I$ is violated because the water level can increase up to extra 300 ms (the analysis took 1.46 seconds).

### 6.3 Networked Thermostat Controllers

A number of rooms are interconnected by open doors, as shown in Fig. 9. The temperature $x_i$ of each room $i$ is separately controlled by its own thermostat controller that turns the heater on and off. That is, $x_i$ depends on the heater’s mode $m \in \{m_{na}, m_{ff}\}$ and the temperatures of the connected rooms, and changes according to the ODEs:

$$\dot{x}_i = \begin{cases} K_i (h_i - ((1 - 2c)x_i + cx_{i-1} + cx_{i+1})) & \text{if } m_i = m_{na} \\ -K_i (1 - 2c)x_i + cx_{i-1} + cx_{i+1}) & \text{if } m_i = m_{ff} \end{cases}$$

In each transition, a controller of room $i$ turns on the heater if $x_i < T_m$, and turns it off if $x_i > T_M$. The safety property is that the temperature $x_i$ of each room is in a certain range $I = [T_m - \eta, T_M + \eta]$, expressed as $(\forall t) x_i(t) \in I$.

We have verified the desired safety property $(\forall t) x_i(t) \in I$ for any number of interconnected thermostat controllers for unbounded time, taking into account clock skews, by compositional inductive reasoning. Provided that both temperatures $x_{i-1}$ and $x_{i+1}$ of the connected rooms are in $I$, we show that $x_i$ is in a tighter range $I' = [T_m - \eta', T_M + \eta] \subseteq I$ with $\eta' < \eta$ during one round. Then, assuming that both $x_{i-1}$ and $x_{i+1}$ are in $I$, we show that $(\forall t) x_i(t) \in I$ is an inductive condition for one round in a similar way to Section 6.2.

We have proved the safety property for any number of thermostats for maximal clock skew $\epsilon = 2$ ms, sampling time $t_s = 10$ ms, and response time $t_r = 200$ ms, using dReal with precision $\delta = 0.001$ (the analysis took 2.6 seconds). However, if $\epsilon = 20$ ms, then the compositional inductive condition $(\forall t) x_i(t) \in I$ is violated because the temperature rises for extra 20 ms and thus $x_i \notin I$ at the end of the round (the analysis took 0.56 seconds).

### 7. CONCLUDING REMARKS

We have presented general techniques for verifying virtually synchronous distributed hybrid systems, with asynchronous communication, imprecise local clocks, network delays, etc., where each component has a local physical environment that can be correlated with other local environments. To make the verification of such systems feasible, we have extended the PALS methodology to hybrid systems, and have given a bisimulation equivalence between the distributed model and the much simpler “synchronous” model, which abstracts from message exchange (and the resulting interleavings), network delays, execution times, etc. However, Hybrid PALS cannot abstract from imprecise local clocks and the timing of sensing and acting.

We have shown that verification problems for Hybrid PALS synchronous models (and, by our bisimulation result, the corresponding distributed hybrid systems) such as bounded reachability analysis, unbounded inductive reasoning, and compositional assume-guarantee reasoning, can be expressed as SMT formulas over the real numbers. We have verified safety properties of a number of non-trivial distributed hybrid systems, with nonlinear ODEs and continuous physical connections between different components, using dReal.

Future work should develop SMT techniques for finding inductive invariant and compositional I/O conditions for nonlinear distributed hybrid systems.

### 8. REFERENCES


