SMT-Based Analysis of Virtually Synchronous Distributed Hybrid Systems*

Kyungmin Bae
SRI International

Peter Csaba Ölveczky
University of Oslo

Soonho Kong
Carnegie Mellon University

Sicun Gao
MIT CSAIL

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.

*This work was partially supported by ONR Grant N000141310090, NSF CPS-1330014 and CPS-1446675, and Air Force STTR Grant F14A-T06-0230.

HSCC'16, April 12-14, 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2883817.2883849

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 SMT encodings for Hybrid PALS models and their analysis. Section 6 gives an overview of the Hybrid PALS case studies. 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 redefined 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, but does *not* present general SMT techniques for analyzing synchronous Hybrid PALS models.

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

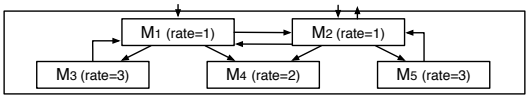

Figure 1: A multirate ensemble $\mathcal{E}_T$.

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) $D_o = D_{o_1} \times \cdots \times D_{o_m}$ an output set, and (iv) $\delta_M \subseteq (D_i \times S) \times (S \times D_o)$ a total transition relation.

A collection $\{M_j\}_{j \in J_S \cup J_F}$ of state machines with different rates can be composed into a *multirate ensemble* $\mathcal{E}_T$ 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 $\mathcal{E}_T$, all components perform a transition *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 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-tuple of inputs for a fast machine.

This *synchronous composition* of an ensemble $\mathcal{E}_T$ is thus equivalent to one machine $M_{\mathcal{E}_T} = (D_i^{\mathcal{E}_T}, S^{\mathcal{E}_T}, D_o^{\mathcal{E}_T}, \delta_{M_{\mathcal{E}_T}})$. If a machine in $\mathcal{E}_T$ 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, $M_{\mathcal{E}_T}$'s states $S^{\mathcal{E}_T}$ consist of the states of its subcomponents M_j and the "feedback" outputs. For example, $M_{\mathcal{E}_T}$ of the ensemble $\mathcal{E}_T$ in Fig. 1 is the machine given by the outer box.

Definition 2. The transition system for $M_{\mathcal{E}_T}$ is a tuple $ts(M_{\mathcal{E}_T}) = (S^{\mathcal{E}_T} \times D_i^{\mathcal{E}_T}, \longrightarrow_{\mathcal{E}_T})$, where $(\vec{s}_1, \vec{i}_1) \longrightarrow_{\mathcal{E}_T} (\vec{s}_2, \vec{i}_2)$ iff an ensemble in state $\vec{s}_1$ with input $\vec{i}_1$ from the interface has a transition to state $\vec{s}_2$ (i.e., $\exists \vec{o}\, ((\vec{i}_1, \vec{s}_1), (\vec{s}_2, \vec{o})) \in \delta_{M_{\mathcal{E}_T}}$).

3.2 PALS Distributed Real-Time Models

Each component in the distributed model $\mathcal{MA}(\mathcal{E}_T, \Gamma)$ is composed of a machine in $\mathcal{E}_T$ and *wrappers* around it, as depicted in Fig. 2. In $\mathcal{MA}(\mathcal{E}_T, \Gamma)$, 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

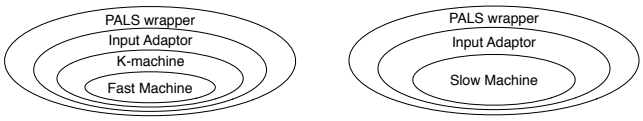

Figure 2: The wrapper hierarchies in $\mathcal{MA}(\mathcal{E}_T, \Gamma)$.

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 $2\epsilon - \mu_{min}$. 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 $\mathcal{MA}(\mathcal{E}_T, \Gamma)$ are snapshots of the system at times $iT-\epsilon$, just before the components in $\mathcal{MA}(\mathcal{E}_T, \Gamma)$ start performing local machine transitions [15]. In $\mathcal{MA}(\mathcal{E}_T, \Gamma)$, 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 full, and the other input and output buffers are empty. The function *sync* maps stable states to the corresponding states of $M_{\mathcal{E}_T}$.

Definition 3. For a stable state C of $\mathcal{MA}(\mathcal{E}_T, \Gamma)$, $sync(C)$ is a pair $(\{s_j, \vec{f_j}\}_{j \in J_S \cup J_F}, \vec{i}) \in S^{\mathcal{E}_T} \times D_i^{\mathcal{E}_T}$ such that: (i) the machine states in C give the states $\{s_j\}_{j \in J_S \cup J_F}$ in $M_{\mathcal{E}_T}$; and (ii) the values in the input buffers of the PALS wrappers in C give the values $\{\vec{f_j}\}_{j \in J_S \cup J_F}$ in the feedback wires and the input $\vec{i}$ from the ensemble interface in $M_{\mathcal{E}_T}$.

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

THEOREM 1. *[3] The binary relation* $(\sim_{obl}; sync)$ *is a* bisimulation *between the transition system* $ts(M_{\mathcal{E}_T})$ *and the big-step transition system* $(Stable(\mathcal{MA}(\mathcal{E}_T, \Gamma)), \longrightarrow_{st})$.

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.

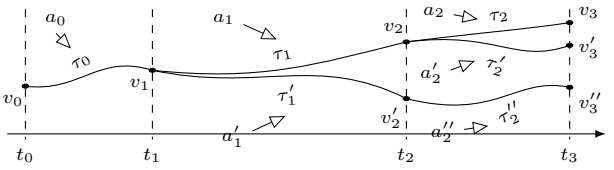

Figure 3: A controlled physical environment; e.g., $((a_1, v_1, t_2 - t_1), \tau_1) \in \Lambda$, $((a_1', v_1, t_2 - t_1), \tau_1') \in \Lambda$, etc.

In Hybrid PALS, the standard PALS models $\mathcal{MA}(\mathcal{E}_T, \Gamma)$ and $\mathcal{E}_T$ are *nondeterministic models* defined for *all possible* environment behaviors. For a physical environment E, the *environment restrictions* $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction E$ and $\mathcal{E}_T \restriction E$ define the behavior of the models constrained by E. Section 4.5 gives a *bisimulation equivalence* between $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction E$ and $\mathcal{E}_T \restriction E$ (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* $\tau_1, \ldots, \tau_l$ of $\vec{x}$ over time. A trajectory of duration T is a function $\tau : [0, T] \to \mathbb{R}$ [14]. The *prefix* of τ at time $u \in [0, T]$ is denoted by $\tau \trianglelefteq u$, and the *suffix* of τ at time u is denoted by $\tau \trianglerighteq u$ (that is, $(\tau \trianglelefteq u)(t) = \tau(t)$ for $t \in [0, u]$, and $(\tau \trianglerighteq u)(t) = \tau(t+u)$ for $t \in [0, T-u]$). Let $\mathcal{T}$ denote the set of all trajectories.

A physical environment E_M of machine M is specified as a *controlled physical environment*, defining any possible trajectory of its 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 physical environment E_M gives a trajectory $\vec{\tau} \in \mathcal{T}^l$ of its parameters $\vec{x}$ of duration t, as illustrated in Fig. 3.

Definition 4. A *controlled physical environment* is a tuple $E_M = (C, \vec{x}, \Lambda)$, 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) $\Lambda \subseteq (C \times \mathbb{R}^l \times \mathbb{R}_{\geq 0}) \times \mathcal{T}^l$ is a *physical transition relation* such that $((a, \vec{v}, t), \vec{\tau}) \in \Lambda$ 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 state $\vec{\tau}(0) = \vec{v} \in \mathbb{R}^l$.

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_1} must be equal to parameter x_2 of E_{M_2}, 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 x_M according to the moving rate r_M (the control command from M) by the

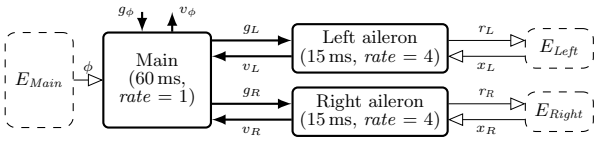

Figure 4: The simple distributed CPS controller.

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 $\Lambda_M \subseteq (\mathbb{R} \times \mathbb{R} \times \mathbb{R}_{\geq 0}) \times \mathcal{T}$ such that $\big((r_M, v_M, 15), \tau\big) \in \Lambda_M$ iff $\forall t \in [0, 15]$. $\tau(t) = v_M + \int_0^t r_M \, \mathrm{d}t$.

The physical environment E_{Main} of the main controller specifies the dynamics of the roll angle ϕ using the ODEs $\dot{\phi} = p$ and $\dot{p} = c(\zeta_R - \zeta_L)$, disregarding the yawing effect caused by the rolling, where p is the rolling moment, and ζ_R and ζ_L are the angles of respective ailerons. The controlled physical environment is $E_{Main} = (\{*\}, (\phi, p, \zeta_L, \zeta_R), \Lambda_{Main})$, with the singleton $\{*\}$, indicating that E_{Main} has no control command, and $\Lambda_{Main} \subseteq (\{*\} \times \mathbb{R}^4 \times \mathbb{R}_{\geq 0}) \times \mathcal{T}^4$ such that $\big((*, (v_\phi, v_p, v_{\zeta_L}, v_{\zeta_R}), 60), (\tau_\phi, \tau_p, \tau_{\zeta_L}, \tau_{\zeta_R})\big) \in \Lambda_{Main}$ iff:

$$\forall t \in [0, 60]. \begin{bmatrix} \tau_\phi \\ \tau_p \end{bmatrix}(t) = \begin{bmatrix} v_\phi \\ v_p \end{bmatrix} + \int_0^t \begin{bmatrix} \tau_p(t) \\ c(\tau_{\zeta_R}(t) - \tau_{\zeta_L}(t)) \end{bmatrix} \mathrm{d}t.$$

The control angles ζ_L and ζ_R must always be the same as the respective aileron angles x_L and x_R of the subcontrollers. However, since the main controller and the subcontrollers have *different periods with local clock skews*, the ODEs of the subcontrollers cannot be "plugged" into E_{Main}. Instead, their physical correlations are specified by the time-invariant constraint: $\forall t. (\zeta_L(t) = x_L(t)) \wedge (\zeta_R(t) = x_R(t))$. ■

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 $\epsilon > 0$. Let $c_M : \mathbb{N} \to \mathbb{R}_{>0}$ 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 $\vec{v}_I$ 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 $\vec{v}_I$, 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 $\vec{o}$ 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 $\vec{\tau}$ 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 $\vec{\tau}$ 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 $\vec{\tau} \in \mathcal{R}^T_{E_M}(a, a', u)$, iff $((a, \vec{\tau}(0), u), \vec{\tau} \trianglelefteq u) \in \Lambda$ and $((a', \vec{\tau}(u), T - u), \vec{\tau} \trianglerighteq u) \in \Lambda$.

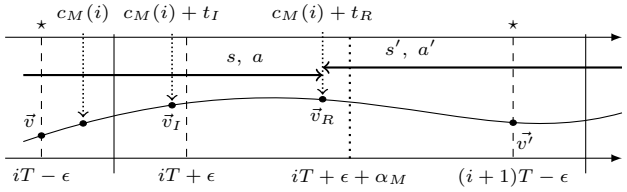

Figure 5: Timeline for an environment-restricted controller with a local clock c_M, where the curved line represents the state of E_M, and the thick straight lines represent the discrete states of M.

In practice, the sampling and response times depend on the machine instructions of a controller M to perform these tasks, which are expressed as state transitions in our model. Therefore, we assume that the sampling times of M depend on its state s, and the response times depend on its state s and input $\vec{i}$. To compose M with its physical environment E_M, we define the "interface" between them.

Definition 6. The *interface* of controller M is defined by the projection functions $\pi = (\pi_C, \pi_T, \pi_R, \pi_I)$, for state s and input $\vec{i}$: (i) $\pi_C(s) \in C$ the control command of M to E_M; (ii) $\pi_T(s) \in \mathbb{N}$ a round number; (iii) $\pi_R(s, \vec{i}) \in \mathbb{R}$ a response time; and (iv) $\pi_I(s) \in \mathbb{R}$ a sampling time ($\pi_I(s) \leq \pi_R(s, \vec{i})$).

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 $S \times \mathbb{R}^m$, where $\mathbb{R}^m$ denotes the m physical parameters that M can *observe*. Likewise, E_M has a state space of the form $\mathbb{R}^l = \mathbb{R}^m \times \mathbb{R}^n$, where $\mathbb{R}^m$ is the *observable* part of $\mathbb{R}^l$. For state $\vec{v} \in \mathbb{R}^l$ of E_M, let $\pi_O(\vec{v}) \in \mathbb{R}^m$ denote the observable part of $\vec{v}$.

The *environment restriction* $M \restriction E_M$ is a normal machine (see Def. 1) with a combined state space $S \times \mathbb{R}^l$, recording "snapshots" of E_M's states at times $iT - \epsilon$. A transition of $M \restriction E_M$ from state $(s, \vec{v})$ to $(s', \vec{v}')$ corresponds to *realizable trajectories* $\vec{\tau}$ for its period T with respect to $\pi_C(s)$ and $\pi_C(s')$ and the response duration $u_R = (c_M(i) + \pi_R(s)) - (iT - \epsilon)$. The controller M performs a transition based on the observable physical state $\pi_O(\vec{\tau}(u_I))$ of E_M sampled after the sampling duration $u_I = (c_M(i) + \pi_I(s)) - (iT - \epsilon)$.

Definition 7. The *environment restriction* of M by E_M with an interface π is $M \restriction_\pi E_M = (D_i, S \times \mathbb{R}^l, D_o, \delta_{M \restriction_\pi E_M})$, where $((\vec{i}, (s, \vec{v})), ((s', \vec{v}'), \vec{o})) \in \delta_{M \restriction_\pi E_M}$ iff for the round numbers $i = \pi_T(s)$ and $\pi_T(s') = i+1$, the sampling duration $u_I = (c_M(i) + \pi_I(s)) - (iT - \epsilon)$, and the response duration $u_R = (c_M(i) + \pi_R(s)) - (iT - \epsilon)$: (i) $\vec{\tau}(0) = \vec{v}$ and $\vec{\tau}(T) = \vec{v}'$ for some realizable trajectories $\vec{\tau} \in \mathcal{R}^T_{E_M}(\pi_C(s), u_R, \pi_C(s'))$, and (ii) $\big((\vec{i}, (s, \pi_O(\vec{\tau}(u_I)))), ((s', \pi_O(\vec{v}')), \vec{o})\big) \in \delta_M$.

Example 2. Consider a subcontroller M in Example 1 to move an aileron toward the specified angle. In each 15 ms round, beginning at time $c_M(i) \in (i \cdot 15\,\mathrm{ms} - \epsilon, i \cdot 15\,\mathrm{ms} + \epsilon)$, M receives a goal angle g_M, sets a new moving rate r'_M based on g_M and the observed angle v_M, and sends back v_M. M is specified as the machine $(\mathbb{R}, \mathbb{N} \times \mathbb{R}^2, \mathbb{R}, \delta_M)$, where $\big((g_M, (i, r_M, v_M)), ((i', r'_M, v'_M), o)\big) \in \delta_M$ holds iff $i' = i + 1$, $r'_M = (g_M - v_M)/15$, and $o = v_m$.

The interface π_M is defined by the projection functions: (i) $\pi_C(i, r_M, v_M) = r_M$; (ii) $\pi_T(i, r_M, v_M) = i$ (the round number); (iii) $\pi_I(i, r_M, v_M) = 1\,\text{ms}$ (the sampling time); and (iv) $\pi_R((i, r_M, v_M), g_M) = 2\,\text{ms}$ (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 $E_M = (\mathbb{R}, x_M, \Lambda_M)$ in Example 1, its observable state is given by the function $\pi_O(x_M) = x_M$.

The environment restriction of M by E_M is then defined as the machine $M \restriction_{\pi_M} E_M = (\mathbb{R}, \mathbb{N} \times \mathbb{R}^2, \mathbb{R}, \delta_{M\restriction_{\pi_M} E_M})$, where $((g_M, (i, r_M, v_M)), ((i', r'_M, v'_M), v_M)) \in \delta_{M\restriction_{\pi_M} E_M}$ holds iff: (i) for some realizable trajectory $\tau \in \mathcal{R}^{15\,\text{ms}}_{E_M}(r_M, r'_M, u_R)$, $v_M = \tau(0)$ and $v'_M = \tau(15)$; and (ii) an M's transition $((g_M, (i, r_M, \tau(u_I))), ((i', r'_M, v'_M), v_M)) \in \delta_M$ holds. ■

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 $c_{Main}(i) \in (i{\cdot}60\,\text{ms}-\epsilon, i{\cdot}60\,\text{ms}+\epsilon)$, M_{Main} receives a desired roll angle g_ϕ 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 $(\mathbb{R}^3, \mathbb{N}\times\mathbb{R}^2, \mathbb{R}^3, \delta_{Main})$, where $\big(((g_\phi, v_L, v_R), (i, v_\phi)), ((i', v'_\phi), (o, g_L, g_R))\big) \in \delta_{Main}$ holds iff: $g_R = 0.3(g_\phi - v_\phi)$, $g_L = -g_R$, $o = v_\phi$, and $i' = i + 1$.

The interface π_{Main} is defined by the projection functions: (i) $\pi_C(i, v_\phi) = *$ (no control commands); (ii) $\pi_T(i, v_\phi) = i$; (iii) $\pi_I(i, v_\phi) = 3\,\text{ms}$; and (iv) $\pi_R((i, v_\phi), \vec{i}) = 10\,\text{ms}$. The sampling duration is $u_I = (c_{Main}(i) + 3\,\text{ms}) - (i \cdot 60\,\text{ms} - \epsilon)$ for each round (the response duration is not important). For E_{Main} in Example 1, since M_{Main} can only observe the roll angle ϕ, its observable state is $\pi_O(\phi, p, \zeta_L, \zeta_R) = \phi$.

The environment restriction of M_{Main} by E_{Main} is then $M_{Main} \restriction_{\pi_{Main}} E_{Main} = (\mathbb{R}^3, \mathbb{N} \times \mathbb{R}^4, \mathbb{R}^3, \delta_{M_{Main}\restriction_{\pi_{Main}} E_{Main}})$. For states $\vec{v} = (v_\phi, v_p, v_{\zeta_L}, v_{\zeta_R})$ and $\vec{v}' = (v'_\phi, v'_p, v'_{\zeta_L}, v'_{\zeta_R})$, a transition $(((g_\phi, v_L, v_R), (i, \vec{v})), ((i', \vec{v}'), (v_\phi, g_L, g_R)))$ holds iff: (i) $\vec{v} = \vec{\tau}(0)$ and $\vec{v}' = \vec{\tau}(60)$ for realizable trajectories $\vec{\tau} \in \mathcal{R}^{60\,\text{ms}}_{E_{Main}}(*, *, 10\,\text{ms})$, and (ii) for $\vec{\tau} = (\tau_\phi, \tau_p, \tau_{\zeta_L}, \tau_{\zeta_R})$, $(((g_\phi, v_L, v_R), (i, \tau_\phi(u_I))), ((i', v'_\phi), (v_\phi, g_L, g_R))) \in \delta_{Main}$. ■

4.4 Hybrid PALS Synchronous Models

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

Definition 8. A *hybrid multirate ensemble* $\mathcal{E}_T \restriction_\Pi E$ is composed of: (i) $\mathcal{E}_T$ a multirate ensemble, (ii) a family of interface functions $\Pi = \{\pi_j\}_{j\in J_S\cup J_F}$, and (iii) a family of local physical environments $E = \langle\{E_{M_j}\}_{j\in J_S\cup J_F}, (\forall t)\,\psi\rangle$ with $(\forall t)\,\psi$ the time-invariant constraints (shown in Fig. 6).

Example 4. For the simple CPS controller in Example 1, $\mathcal{E}_{60}$ 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 constraint $(\forall t)\,\psi \equiv (\forall t)\,(\zeta_L(t) = x_L(t)) \wedge (\zeta_R(t) = x_R(t))$ specifies the continuous physical connections. ■

A hybrid ensemble $\mathcal{E}_T \restriction_\Pi E$ induces a normal multirate ensemble $\overline{\mathcal{E}_T \restriction_\Pi E}$ composed of the environment restrictions $\{M_j \restriction_\pi E_{M_j}\}_{j\in J_S\cup J_F}$, which disregards the time-invariant constraints $(\forall t)\,\psi$. The behaviors of $\mathcal{E}_T \restriction_\Pi E$ are a *subset* of the behaviors of $\overline{\mathcal{E}_T \restriction_\Pi E}$, namely, the behaviors restricted by $(\forall t)\,\psi$. As mentioned, a transition of a (decelerated) environment-restricted machine $M_j \restriction_\pi E_{M_j}$ corresponds to realizable trajectories $\vec{\tau}$ 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* $\vec{\tau}$ *also satisfy the time-invariant constraints* $(\forall t)\,\psi$ gives a transition of the synchronous composition $M_{\mathcal{E}_T\restriction_\Pi E}$.

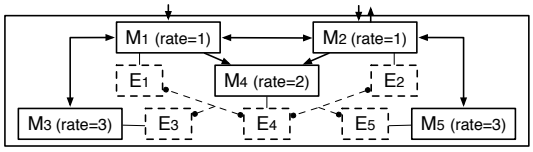

Figure 6: A hybrid multirate ensemble $\mathcal{E}_T \restriction_\Pi E$, where (E_1, E_3, E_4) and (E_2, E_4, E_5) are connected by time-invariant constraints, denoted by dashed lines.

4.5 Hybrid PALS Distributed Models

Hybrid PALS maps a hybrid ensemble $\mathcal{E}_T \restriction_\Pi E$ to the distributed hybrid system $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$, together with PALS bounds Γ; i.e., $(\mathcal{E}_T \restriction_\Pi E, \Gamma) \mapsto \mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$. The distributed components in $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ are exactly the same as the wrapper hierarchies of Fig. 2 in $\mathcal{MA}(\mathcal{E}_T, \Gamma)$. But the controllers in $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ also interact with their physical environments in E, according to the sampling and response timing policy Π to determine sensor sampling timing t_I and actuator response timing t_R.

The behaviors of the Hybrid PALS distributed system $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ are the subset of those of the PALS distributed system $\mathcal{MA}(\mathcal{E}_T, \Gamma)$, restricted by the physical environments and the time-invariant constraints in E. The continuous dynamics of $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ 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 $\overline{\mathcal{E}_T \restriction_\Pi E}$ above, induced by a hybrid ensemble $\overline{\mathcal{E}_T \restriction_\Pi E}$, and composed of the environment-restricted controllers $\{M_j \restriction_\pi E_{M_j}\}_{j\in J_S\cup J_F}$. 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 $\mathcal{MA}(\overline{\mathcal{E}_T \restriction_\Pi E}, \Gamma)$, as explained in Section 3.2. For such time intervals $[iT - \epsilon, (i + 1)T - \epsilon]$, each $M_j \restriction_\pi E_{M_j}$ provides (k-step) realizable trajectories $\vec{\tau}_j$, based on round numbers, control commands, sampling timings, and response timings. The behaviors of $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ are the behaviors of $\mathcal{MA}(\overline{\mathcal{E}_T \restriction_\Pi E}, \Gamma)$ that are restricted by the time-invariant constraints. Big-step transitions of $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ are therefore exactly those of $\mathcal{MA}(\overline{\mathcal{E}_T \restriction_\Pi E}, \Gamma)$ whose associated trajectories $\vec{\tau}_j$ satisfy the time-invariant constraints $(\forall t)\,\psi$.

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

THEOREM 2. *The relation* $(\sim_{obi}; sync)$ *is a bisimulation between the transition system* $ts(M_{\mathcal{E}_T\restriction_\Pi E})$ *and the big-step transition system induced by* $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$*, exhibiting the exactly same set of realizable trajectories.*

PROOF SKETCH. Suppose that there exists a transition $s \longrightarrow_{\mathcal{E}_T\restriction_\Pi E} s'$ of $M_{\mathcal{E}_T\restriction_\Pi E}$ and s $(\sim_{obi}; sync)$ C for a stable

state C of $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$. By definition, there exists $s \longrightarrow_{\overline{\mathcal{E}_T \restriction_\Pi E}} s'$ of $M_{\overline{\mathcal{E}_T \restriction_\Pi E}}$ with some realizable trajectories $\vec{\tau}$ that satisfy the time-invariant constraints $(\forall t)\, \psi$ for the induced ensemble $\overline{\mathcal{E}_T \restriction_\Pi E}$. By Theorem 1, $(\sim_{obi}; \mathit{sync})$ is a bisimulation between the transition systems $ts(M_{\overline{\mathcal{E}_T \restriction_\Pi E}})$ and $(\mathit{Stable}(\mathcal{MA}(\overline{\mathcal{E}_T \restriction_\Pi E}, \Gamma)), \longrightarrow_{st})$. Therefore, for some stable state C', there is a big-step transition $C \longrightarrow_{st} C'$ in $\mathcal{MA}(\overline{\mathcal{E}_T \restriction_\Pi E}, \Gamma)$ such that s' $(\sim_{obi}; \mathit{sync})$ C', where both $s \longrightarrow_{\overline{\mathcal{E}_T \restriction_\Pi E}} s'$ and $C \longrightarrow_{st} C'$ 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 $\vec{\tau}$, which satisfy $(\forall t)\, \psi$, are also realizable for $C \longrightarrow_{st} C'$, and there exists a big-step transition $C \longrightarrow_{st} C'$ by $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$. The other direction is similar. □

5. HYBRID PALS MODELS IN SMT

Theorem 2 implies that analyzing $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ can be reduced to the much simpler problem of analyzing $M_{\mathcal{E}_T \restriction_\Pi E}$, which abstracts from asynchronous communication, network delays, execution times, message buffering, etc. This section shows how analysis problems for a hybrid ensemble $\mathcal{E}_T \restriction_\Pi E$ can be encoded as formulas over the real numbers and ODEs. We symbolically encode *all possible local clocks* to deal with clock skews. Standard formal analysis problems for hybrid systems, such as bounded reachability, inductive reasoning, and compositional assume-guarantee reasoning, are encoded as logical formulas. The satisfiability of such formulas can be decided by δ-complete SMT solving [8, 10] up to a given precision $\delta > 0$. δ-complete SMT solving for a formula ϕ returns false if ϕ is unsatisfiable, and returns true if its *syntactic numerical perturbation* of ϕ by δ is satisfiable.[1]

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 $\phi_M(\vec{i}, \vec{y} \mid \vec{y'}, \vec{o})$, with variables $\vec{i}$, $\vec{y}$, $\vec{y'}$, and $\vec{o}$ denoting input, the current state, the next state, and output, respectively, in such a way that $\phi_M(\vec{i}, s \mid s', \vec{o}) \iff ((\vec{i}, s), (s', \vec{o})) \in \delta_M$.

Example 5. For an aileron subcontroller M in Example 2, the formula is $\phi_M(y_{g_M}, y_i, y_{r_M}, y_{v_M} \mid y'_i, y'_{r_M}, y'_{v_M}, y_o) \equiv (y'_{r_M} = (y_{g_M} - y_{v_M})/15 \wedge y_o = y_{v_M} \wedge y'_i = y_i + 1)$. Likewise, the formula for the main controller M_{Main} in Example 3 is $\phi_{Main}(y_{g_\phi}, y_{v_L}, y_{v_R}, y_i, y_{v_\phi} \mid y'_i, y'_{v_\phi}, y_o, y_{g_L}, y_{g_R}) \equiv (y_{g_R} = 0.3(y_{g_\phi} - y_{v_\phi}) \wedge y_{g_L} = -y_{g_R} \wedge y_o = y_{v_\phi} \wedge y'_i = y_i + 1)$. ■

A physical environment E_M is encoded as a formula of the form $\phi_{E_M}(\vec{a}, \vec{v}, u_0, u_t \mid \vec{\tau})$, with: *unary function symbols* $\vec{\tau}$ denoting the trajectories of E_M's physical parameters $\vec{x}$, and *variables* $\vec{a}$, $\vec{v}$, u_0, and u_t denoting, respectively, control commands, the initial values of $\vec{\tau}$ at time u_0, the times at the beginning and the end of the trajectory duration, where $\phi_{E_M}(\vec{a}, \vec{v}, u_0, u_t \mid \vec{\tau}) \iff ((\vec{a}, \vec{v}, u_t - u_0), \vec{\tau} \trianglerighteq u_0) \in \Lambda$.

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 $\vec{a}$ and an interval $[u_0, u_t]$, then ϕ_{E_M} includes universal quantification over time along with solutions of the ODEs, for example: $\bigvee \left(\mathit{guard}(\vec{a}) \to \forall t \in [u_0, u_t].\ \vec{\tau}(t) = \vec{v} + \int_0^{t-u_0} F_{\vec{a}}(\vec{x}, t)\, \mathrm{d}t\right)$.

[1] If $\phi \equiv (x > 3) \wedge (y = z)$, then its syntactic numerical perturbation by δ is $(x - 3 > -\delta) \wedge (y - z \geq -\delta) \wedge (z - y \geq -\delta)$.

Example 6. For an aileron subcontroller M in Example 1, the formula $\phi_{E_M}(y_{r_M}, y_{v_M}, y_{u_0}, y_{u_t} \mid \tau_M)$ is defined by:

$$\forall t \in [y_{u_0}, y_{u_t}].\ \tau_M(t) = y_{v_M} + \int_0^{t - y_{u_0}} y_{r_M}\, \mathrm{d}t.$$

For the main controller *Main*, if $y_{u_0} = 0$, then the formula $\phi_{E_{Main}}(y_{v_\phi}, y_{v_p}, y_{v_{\zeta_L}}, y_{v_{\zeta_R}}, 0, y_{u_t} \mid \tau_\phi, \tau_p, \tau_{\zeta_L}, \tau_{\zeta_R})$ is:

$$\forall t \in [0, y_{u_t}]. \begin{bmatrix} \tau_\phi(t) \\ \tau_p(t) \end{bmatrix} = \begin{bmatrix} y_{v_\phi} \\ y_{v_p} \end{bmatrix} + \int_0^t \begin{bmatrix} \tau_p(t) \\ c(\tau_{\zeta_R}(t) - \tau_{\zeta_L}(t)) \end{bmatrix} \mathrm{d}t,$$

where τ_{ζ_R} and τ_{ζ_L} are given by the subcontrollers using the time-invariant constraints. ■

The encoding of an environment restriction $M \restriction_\pi E_M$ has the form $\phi^{T,i}_{M \restriction_\pi E_M}(\vec{i}, \vec{y}, \vec{v} \mid \vec{y'}, \vec{v'}, \vec{o} \mid \vec{\tau})$, with unary function symbols $\vec{\tau}$ denoting E_M's trajectories, and 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, given a period T and a round number i.

The values of sampling duration $u_I = (c_M(i) + t_I) - (iT - \epsilon)$ and response duration $u_R = (c_M(i) + t_R) - (iT - \epsilon)$ are unknown, due to clock skews. Since $iT - \epsilon < c_M(i) < iT + \epsilon$, we represent those times as formulas $t_I < u_I < t_I + 2\epsilon$ and $t_R < u_R < t_R + 2\epsilon$, so that $\phi_{M \restriction_\pi E_M}(\vec{i}, s, \vec{v} \mid s', \vec{v'}, \vec{o})$ iff $((\vec{i}, (s, \vec{v})), ((s', \vec{v'}), \vec{o})) \in \delta_{M \restriction_\pi E_M}$ for *some* local clock c_M.

For an environment restriction $M \restriction_\pi E_M$, the formula $\phi^{T,i}_{M \restriction_\pi E_M}(\vec{i}, \vec{y}, \vec{v} \mid \vec{y'}, \vec{v'}, \vec{o} \mid \vec{\tau})$ is defined as follows (from the formal definitions of $M \restriction_\pi E_M$ in Definitions 5–7):

$$\begin{aligned}
(\exists \vec{a}, \vec{a'}, u_I, u_R, \vec{v}_I, \vec{v}_R)\ \ & \vec{a} = \pi_C(\vec{y}) \ \wedge\ \vec{a'} = \pi_C(\vec{y'}) \ \wedge \\
& \pi_I(\vec{y}) < u_I < \pi_I(\vec{y}) + 2\epsilon \ \wedge\ \vec{v}_I = \vec{\tau}(u_I) \ \wedge \\
& \pi_R(\vec{y}, \vec{i}) < u_R < \pi_R(\vec{y}, \vec{i}) + 2\epsilon \ \wedge\ \vec{v}_R = \vec{\tau}(u_R) \ \wedge \\
& \phi_{E_M}(\vec{a}, \vec{v}, iT, iT + u_R \mid \vec{\tau}) \ \wedge\ \vec{v} = \vec{\tau}(0) \ \wedge \\
& \phi_{E_M}(\vec{a'}, \vec{v}_R, iT + u_R, (i+1)T, \mid \vec{\tau}) \ \wedge\ \vec{v'} = \vec{\tau}(T) \ \wedge \\
& \phi_M(\vec{i}, \langle \vec{y}, \pi_O(\vec{\tau}(u_I)) \rangle \mid \langle \vec{y'}, \pi_O(\vec{v'}) \rangle, \vec{o}).
\end{aligned}$$

Example 7. For $M \restriction_\pi E_M$ in Example 2, the formula $\phi^{15,0}_{M \restriction_\pi E_M}(y_{g_M}, y_i, y_{r_M}, y_{v_M} \mid y'_i, y'_{r_M}, y'_{v_M}, y_o \mid \tau_M)$ is given by: $\exists u_I, u_R, v_I, v_R.\ (1 < u_I < 1 + 2\epsilon) \wedge (2 < u_R < 2 + 2\epsilon) \wedge v_I = \tau_M(u_I) \wedge v_R = \tau_M(u_R) \wedge \phi_{E_M}(y_{r_M}, y_{v_M}, 0, u_R \mid \tau_M) \wedge \phi_{E_M}(y'_{r_M}, v_R, u_R, 15, \mid \tau_M) \wedge y_{v_M} = \tau_M(0) \wedge y'_{v_M} = \tau_M(15) \wedge \phi_M(y_{g_M}, y_i, y_{r_M}, v_I \mid y'_i, y'_{r_M}, y'_{v_M}, y_o)$. ■

5.1.2 Encoding Hybrid Ensembles

Recall that all machines in a synchronous composition $M_{\mathcal{E}_T}$ perform their transitions in lockstep; for a fast machine f, $k = \mathit{rate}(f)$ transitions in a global round. The encoding $\phi^{T,i}_{(M \restriction_\pi E_M)^{\times k}}$ of the k-step *deceleration* of an environment restriction $M \restriction_\pi E_M$ is given by sequentially composing the k formulas $\phi^{T/k,ik}_{M \restriction_\pi E_M}, \ldots, \phi^{T/k,(ik+k-1)}_{M \restriction_\pi E_M}$ for the k *subintervals* $[(ik + n - 1)T/k - \epsilon, (ik + n)T/k - \epsilon]$ for $n = 1, \ldots, k$.

The encoding ϕ_{wire} of $\mathcal{E}_T$'s wiring diagram is a conjunction of equalities between variables for input and output ports. Each equality corresponds to a connection in $\mathcal{E}_T$ (together with an input adaptor for machines with different rates). Since feedback outputs becomes input of their destinations in the next step, we use a separate set of variables for such output ports connected to machines in $\mathcal{E}_T$.

A hybrid ensemble $\mathcal{E}_T \restriction_\Pi E$ of typed machines $\{M_j\}_{j\in J}$, physical environments $\{E_{M_j}\}_{j\in J}$, for $J = J_S \cup J_F$, and the time-invariant constraint $(\forall t.\ \psi)$ is encoded as a formula $\phi_{\mathcal{E}_T \restriction_\Pi E}(\vec{i}, \{\vec{y_j}, \vec{v_j}, \vec{f_j}\}_{j\in J} \mid \{\vec{y'_j}, \vec{v'_j}, \vec{f'_j}\}_{j\in J}, \vec{o} \mid \{\vec{\tau_j}\}_{j\in J})$ is:

$$\exists\{\vec{i_j}, \vec{o_j}\}_{j\in J}.\ \textstyle\bigwedge_{s\in J_S} \left(\phi^{T,0}_{M_s \restriction_{\pi_s} E_{M_s}}(\vec{i_s}, \vec{y_s}, \vec{v_s} \mid \vec{y'_s}, \vec{v'_s}, \vec{o_s} \mid \vec{\tau_s})\right)$$
$$\wedge \textstyle\bigwedge_{f\in J_F} \left(\phi^{T,0}_{(M_f \restriction_{\pi_f} E_{M_f})^{\times rate(f)}}(\vec{i_f}, \vec{y_f}, \vec{v_f} \mid \vec{y'_f}, \vec{v'_f}, \vec{o_f} \mid \vec{\tau_f})\right)$$
$$\wedge\ \phi_{wire}(\vec{i}, \vec{o}, \{\vec{i_j}, \vec{o_j}, \vec{f_j}, \vec{f'_j}\}_{j\in J}) \ \wedge\ (\forall t.\ \psi),$$

with unary function symbols $\vec{\tau_j}$ denoting trajectories for E_{M_j}, and variables $(\vec{y_j}, \vec{v_j})$, $\vec{f_j}$, $(\vec{y'_j}, \vec{v'_j})$, and $\vec{f'_j}$ denoting, respectively, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the beginning of the round, the feedback outputs from the previous round, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the end of the round, and the feedback outputs for the next round. The variable $\vec{i}$ denotes ensemble inputs, and $\vec{o}$ denotes ensemble outputs.

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

THEOREM 3. *The formula $\phi_{\mathcal{E}_T \restriction_\Pi E}$ is satisfiable iff there exists a corresponding stable transition for some local clocks in a distributed hybrid system $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$.*

Example 8. The hybrid ensemble $\mathcal{E}_{60} \restriction_\Pi E$ in Example 4 is encoded as the formula $\phi_{\mathcal{E}_{60} \restriction_\Pi E}$, the conjunction of the following formulas: (i) for the main controller:

$$\phi^{60,0}_{M_M \restriction_{\pi_M} E_M}(y^M_{g_\phi}, y^M_{v_L}, y^M_{v_R}, y_i, \vec{y} \mid y'_i, \vec{y'}, y^M_o, y^M_{g_L}, y^M_{g_R} \mid \vec{\tau})$$

with $\vec{y} = (y_{v_\phi}, y_{v_p}, y_{v_{\zeta_L}}, y_{v_{\zeta_R}})$, $\vec{y'} = (y'_{v_\phi}, y'_{v_p}, y'_{v_{\zeta_L}}, y'_{v_{\zeta_R}})$, and $\vec{\tau} = (\tau_\phi, \tau_p, \tau_{\zeta_L}, \tau_{\zeta_R})$; (ii) for the subcontrollers:

$$\phi^{60,0}_{(M_L \restriction_\pi E_L)^{\times 4}}(\vec{y}_{g_L}, \vec{y}^L_0, y^0_{v_L} \mid \vec{y}^L_4, y^4_{v_L}, \vec{y}_{o_L} \mid \tau_L)$$
$$\wedge\ \phi^{60,0}_{(M_R \restriction_\pi E_R)^{\times 4}}(\vec{y}_{g_R}, \vec{y}^R_0, y^0_{v_R} \mid \vec{y}^R_4, y^4_{v_R}, \vec{y}_{o_R} \mid \tau_R)$$

with $\vec{y}_{g_m} = (y^1_{g_m}, \ldots, y^4_{g_m})$, $\vec{y}_{o_m} = (y^1_{o_m}, \ldots, y^4_{o_m})$, and for $m \in \{L, R\}$, $\vec{y}^m_n = (y^n_{i_m}, y^n_{r_m})$; (iii) for the wiring diagram:

$$(y^M_{v_L} = f^4_{o_L} \wedge f^{4\prime}_{o_L} = y^4_{o_L}) \wedge (\textstyle\bigwedge^4_{i=1} y^i_{g_L} = f^M_{g_L} \wedge f^{M\ \prime}_{g_L} = y^M_{g_L})$$
$$\wedge\ (y^M_{v_R} = f^4_{o_R} \wedge f^{4\prime}_{o_R} = y^4_{o_R}) \wedge (\textstyle\bigwedge^4_{i=1} y^i_{g_R} = f^M_{g_R} \wedge f^{M\ \prime}_{g_R} = y^M_{g_R})$$

with variables f_{port} denoting a feedback output from the previous step, and f'_{port} denoting one for the next step; and, finally, (iv) for the time-invariant equality constraint $(\forall t.\ \tau_{\zeta_L}(t) = \tau_L(t) \ \wedge\ \tau_{\zeta_R}(t) = \tau_R(t))$. ■

5.2 Encoding Verification Problems

Our goal is to verify safety properties of a distributed hybrid model $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$, expressed as formulas of the form $safe(\vec{y}, \vec{\tau}(t))$ for *state variables* $\vec{y}$, *trajectories* $\vec{\tau}$, and time variable t. We exploit the bisimulation equivalence $M_{\mathcal{E}_T \restriction_\Pi E} \approx \mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ to verify $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$ using the simpler synchronous hybrid model $M_{\mathcal{E}_T \restriction_\Pi E}$.

5.2.1 Bounded Reachability

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

Definition 9. Bounded reachability of $M_{\mathcal{E}_T \restriction_\Pi E}$ *up to* n *rounds* for a safety property $safe(\vec{y}, \vec{\tau}(t))$, an initial condition $init(\vec{y})$, and an input constraint $in(\vec{i})$ is encoded by:

$$\exists \vec{y}_0, \{\vec{y}_k, \vec{i}_k, \vec{o}_k, t_k\}^n_{k=1}.\ init(\vec{y}_0) \ \wedge\ \textstyle\bigvee^n_{k=1} \neg safe(\vec{y}_k, \vec{\tau}_k(t_k))$$
$$\wedge\ \textstyle\bigwedge^n_{k=1} \left(\phi_{\mathcal{E}_T \restriction_\Pi E}(\vec{i}_k, \vec{y}_{k-1} \mid \vec{y}_k, \vec{o}_k \mid \vec{\tau}_k) \wedge in(\vec{i}_k)\right).$$

Some initial state $\vec{y}_0$ satisfies the *init* condition, and then $M_{\mathcal{E}_T \restriction_\Pi E}$ performs n steps of synchronous transitions, each of which is from some state $\vec{y}_{k-1}$ to $\vec{y}_k$ using trajectories $\vec{\tau}_k$ of duration T with some input $\vec{i}_k$ satisfying the input constraint *in*. $M_{\mathcal{E}_T \restriction_\Pi E}$ has bounded counterexamples if some state $(\vec{y}_k, \vec{\tau}_k(t_k))$ at some time t_k violates the safety *safe*.

5.2.2 Inductive Reasoning

For *unbounded* time verification, we encode an inductive proof of a safety property $safe(\vec{y}, \vec{\tau}(t))$ as a logical formula by using an inductive condition for $M_{\mathcal{E}_T \restriction_\Pi E}$'s synchronous transitions, which implies the safety property. An inductive invariant consists of two formulas: $ind_d(\vec{y})$ for state variables $\vec{y}$, and $\forall t \in [0, T].\ ind_c(\vec{\tau}(t))$ for trajectories $\vec{\tau}$.[2]

Definition 10. Inductive reasoning of $M_{\mathcal{E}_T \restriction_\Pi E}$ for a safety property $safe(\vec{y}, \vec{\tau}(t))$, an initial condition $init(\vec{y})$, and an input constraint $in(\vec{i})$, using an inductive invariant condition $(ind_d(\vec{y}) \ \wedge\ \forall t \in [0, T].\ ind_c(\vec{\tau}(t)))$ is encoded by:

- $\forall \vec{y}.\ init(\vec{y}) \implies ind_d(\vec{y})$
- $\forall \vec{y}, \vec{y'}, \vec{i}, \vec{o}.\ (ind_d(\vec{y}) \wedge \phi_{\mathcal{E}_T \restriction_\Pi E}(\vec{i}, \vec{y} \mid \vec{y'}, \vec{o} \mid \vec{\tau}) \wedge in(\vec{i})) \implies (ind_d(\vec{y'}) \wedge \forall t \in [0, T].\ ind_c(\vec{\tau}(t)))$
- $\forall \vec{y}, \forall t \in [0, T].\ ind_d(\vec{y}) \wedge ind_c(\vec{\tau}(t)) \implies safe(\vec{y}, \vec{\tau}(t))$

The *init* condition implies the ind_d condition for any state variables $\vec{y}$. If a transition of $M_{\mathcal{E}_T \restriction_\Pi E}$ is taken from state $\vec{y}$ satisfying the ind_d condition, then the ind_d condition again holds for any next state $\vec{y'}$, and in the meantime the ind_c condition holds for trajectories $\vec{\tau}$. The inductive condition $ind_d(\vec{y}) \wedge ind_c(\vec{\tau}(t))$ implies the safety property $safe(\vec{y}, \vec{\tau}(t))$ for one round. By proving these conditions, we show that the safety property holds for unbounded time with unbounded number of transitions. These formulas can be proved by checking the unsatisfiability of their *negated versions.*

5.2.3 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 $M_{\mathcal{E}_T \restriction_\Pi E}$, each component $M_j \restriction_{\pi_j} E_{M_j}$ performs a transition based on its input $\vec{i}_j$ and trajectories $\vec{\tau}_j$ that are restricted by time-invariant constraints. Hence, we use an input condition $c^j_{in}(\vec{i}_j, \vec{\tau}_j(t))$ and an output condition $c^j_{out}(\vec{o}_j, \vec{\tau}_j(t))$, which satisfy necessary constraints for compositional reasoning: (i) assuming an input condition $c^j_{in}(\vec{i}_j, \vec{\tau}_j(t))$, if a transition is taken, then the output condition $c^j_{out}(\vec{o}_j, \vec{\tau}_j(t))$ holds; and (ii) the collection of the output conditions $\{c^j_{out}(\vec{o}, \vec{\tau}_j(t))\}_j$ implies each input condition $c^j_{in}(\vec{i}_j, \vec{\tau}_j(t))$. Let $\overline{M}_j$ denote its decelerated version $(M_j \restriction_{\pi_j} E_{M_j})^{\times rate(j)}$.

[2]A key problem is finding such an inductive invariant for $M_{\mathcal{E}_T \restriction_\Pi E}$, but providing general solutions for this problem is beyond the scope of this paper.

Definition 11. For each component j in a hybrid ensemble $\mathcal{E}_T \restriction_\Pi E$, consider a safety property $safe_j(\vec{y}_j, \vec{\tau}_j(t))$, an input constraint $in_j(\vec{i}_j)$ for input from $\mathcal{E}_T$'s interface, and an initial condition $init_j(\vec{y}_j)$. *I/O conditions* are given by:

$$\forall t.\ c^j_{in}(\vec{i}_j, \vec{\tau}_j(t)) \ \equiv\ (c^j_{in,d}(\vec{i}_j) \ \wedge\ \forall t \in [0,T].\ c^j_{in,c}(\vec{\tau}_j(t)))$$

$$\forall t.\ c^j_{out}(\vec{o}_j, \vec{\tau}_j(t)) \ \equiv\ (c^j_{out,d}(\vec{i}_j) \ \wedge\ \forall t \in [0,T].\ c^j_{out,c}(\vec{\tau}_j(t)))$$

that satisfy the necessary constraints for a compositional reasoning of the synchronous composition $M_{\mathcal{E}_T \restriction_\Pi E}$:

- $\forall \vec{i}_j, \vec{o}_j, \vec{y}_j, \vec{y}'_j.\ \big(\forall t \in [0,T].\ c^j_{in}(\vec{i}_j, \vec{\tau}_j(t)) \wedge \phi^{T,0}_{M_j}(\vec{i}_j, \vec{y}_j \restriction \vec{y}'_j, \vec{o}_j \restriction \vec{\tau}_j)\big) \implies \forall t \in [0,T].\ c^j_{out}(\vec{o}_j, \vec{\tau}_j(t))$
- $\forall t \in [0,T].\ \bigwedge_j c^j_{out}(\vec{o}, \vec{\tau}_j(t)) \implies \bigwedge_j c^j_{in}(\vec{i}_j, \vec{\tau}_j(t))$

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:

$$\exists \vec{y}^j_0, \{\vec{y}^j_k, \vec{i}^j_k, \vec{o}^j_k, t^j_k\}_{k=1}^n.\ init_j(\vec{y}^j_0) \ \wedge\ \bigvee_{k=1}^{n} \neg safe_j(\vec{y}^j_k, \vec{\tau}^j_k(t_k)) \ \wedge\ \bigwedge_{k=1}^{n} \begin{bmatrix} \phi^T_{M_j}(\vec{i}^j_k, \vec{y}^j_{k-1} \restriction \vec{y}^j_k, \vec{o}^j_k \restriction \vec{\tau}^j_k) \ \wedge in_j(\vec{i}^j_k) \\ \wedge \forall t \in [0,T].\ c^j_{in}(\vec{i}^j_k, \vec{\tau}^j_k(t)) \end{bmatrix}$$

where the input condition $\forall t \in [0,T].\ c^j_{in}(\vec{i}^j_k, \vec{\tau}^j_k(t))$ is also assumed for each step in addition to Definition 9.

Likewise, a compositional inductive reasoning problem of $safe_j(\vec{y}_j, \vec{\tau}_j(t))$ for component j with an inductive condition $ind^j_d(\vec{y}_j) \wedge \forall t \in [0,T].\ ind^j_c(\vec{\tau}_j(t))$ can be encoded by:

- $\forall \vec{y}_j.\ init_j(\vec{y}_j) \implies ind^j_d(\vec{y}_j)$
- $\forall \vec{y}_j, \vec{y}'_j, \vec{i}_j, \vec{o}_j.\ \begin{bmatrix} ind^j_d(\vec{y}_j) \ \wedge\ \phi^{T,0}_{M_j}(\vec{i}_j, \vec{y}_j \restriction \vec{y}'_j, \vec{o}_j \restriction \vec{\tau}_j) \wedge \\ in_j(\vec{i}_j) \ \wedge\ \forall t \in [0,T].\ c^j_{in}(\vec{i}_j, \vec{\tau}_j(t)) \end{bmatrix} \implies (ind^j_d(\vec{y}'_j) \ \wedge\ \forall t \in [0,T].\ ind^j_c(\vec{\tau}_j(t)))$
- $\forall \vec{y}_j, \forall t.\ ind^j_d(\vec{y}_j) \wedge ind^j_c(\vec{\tau}_j(t)) \implies safe_j(\vec{y}_j, \vec{\tau}_j(t))$

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 *uninterpreted functions on the real numbers*, such as time-invariant constraints ($\forall t.\ \psi$) or formulas of the form $\forall t \in [u_0, u_t].\ \vec{\tau}(t) = \vec{v} + \int_0^{t-u_0} F_{\vec{a}}(\vec{x}, t)\,\mathrm{d}t$, 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 formulas.

We restrict our attention to time-invariant constraints with only *equality* terms, since continuous correlations typically can be expressed using only equalities (e.g., Example 1). Equality constraints, such as $x_1(t) = x_2(t)$, can be removed from the formula by replacing one side with the other, e.g., by replacing each function symbol x_1 with x_2. From now on we assume that time-invariant equality constraints have been removed from the formula in this way.

[3]Providing general solutions for how to find I/O conditions for $M_{\mathcal{E}_T \restriction_\Pi E}$ is beyond the scope of this paper.

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

Example 9. The physical environment E_{Main} of the main controller in Example 1 involves the formula

$$\forall t \in [0, 60].\ \begin{bmatrix} \tau_\phi(t) \\ \tau_p(t) \end{bmatrix} = \begin{bmatrix} v_\phi \\ v_p \end{bmatrix} + \int_0^t \begin{bmatrix} \tau_p(t) \\ c(\tau_{\zeta_R}(t) - \tau_{\zeta_L}(t)) \end{bmatrix} \mathrm{d}t.$$

including two uninterpreted function symbols τ_{ζ_L} and $\tau\zeta_R$. Consider two logical formulas: for the left subcontroller, $\forall t \in [0, u^L_R].\ \tau_L(t) = v_L + \int_0^t rate^i_L\,\mathrm{d}t$, and for the right subcontroller, $\forall t \in [u^R_R, 15].\ \tau(t) = v_R + \int_0^t rate^r_R\,\mathrm{d}t$. For the common interval $[u^R_R, u^L_R]$, the complete ODE system is:

$$\begin{bmatrix} \tau_\phi(u^L_R) \\ \tau_p(u^L_R) \\ \tau_L(u^L_R) \\ \tau_R(u^L_R) \end{bmatrix} = \begin{bmatrix} \tau_\phi(u^R_R) \\ \tau_p(u^R_R) \\ \tau_L(u^R_R) \\ v_R \end{bmatrix} + \int_0^{u^L_R - u^R_R} \begin{bmatrix} \tau_p(t) \\ c(\tau_{\zeta_R}(t) - \tau_{\zeta_L}(t)) \\ rate^i_L \\ rate^r_R \end{bmatrix} \mathrm{d}t.$$

This system indicates a period that the right subcontroller has responded (with the new rate $rate^r_R$) 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 $\mathcal{E}_T \restriction_\Pi E$ 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 $\forall t \in [u_0, u_t].\ \vec{x}(t) = \vec{v} + \int_0^{t-u_0} F_{\vec{a}}(\vec{x}, t)\,\mathrm{d}t$ in the formula $\phi_{\mathcal{E}_T \restriction_\Pi E}$. 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 $\mathcal{E}_T \restriction_\Pi E$ instead of analyzing the distributed hybrid models $\mathcal{MA}(\mathcal{E}_T, \Gamma) \restriction_\Pi E$.

We have verified safety properties using inductive and compositional SMT encodings for *any possible* set of local clocks with maximal clock skew ϵ. 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.

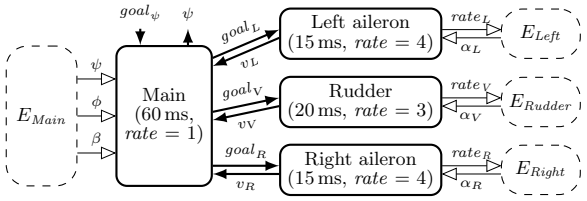

Figure 7: The controllers for turning an airplane.

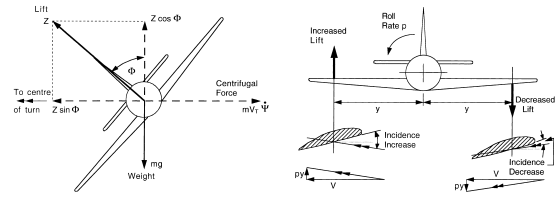

Figure 8: Forces acting in a turn of an aircraft [7].

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 β is always close to 0.

Each subcontroller M gradually moves its surface towards the goal angle $goal_M$ specified by the main controller M_{Main}, as explained in Example 1. In each round, M receives $goal_M$ from M_{Main}, determines the moving rate $rate_M$ based on $goal_M$ and the current sampled value v_M of the angle α_M, and sends back v_M to M_{Main}.[4] The local environment E_M specifies the dynamics of α_M by the ODE $\dot{\alpha_M} = rate_M$.

The main controller M_{Main} determines the goal angles for the subcontrollers to make a coordinated turn. In each round, M_{Main} receives a desired direction $goal_\psi$ (from the pilot) and the angles (v_L, v_V, v_R) from the subcontrollers, and sends back the new goals $(goal_L, goal_V, goal_R)$, based on the current sampled *position* values $(v_\psi, v_\phi, v_\beta)$ of the direction angle ψ, the roll angle ϕ, and the yaw angle β. We use a simple control logic to decide the new goal angles based on the current position angles, namely, by using some function $(goal_L, goal_V, goal_R) = f_{Main}(v_\psi, v_\phi, v_\beta)$.

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

$$\dot{\beta} = Y_{\zeta_L,\zeta_V,\zeta_R,\beta}/mV - r + (g/V)\cos\beta\sin\phi,$$
$$\dot{\phi} = p, \qquad \dot{\psi} = (g/V)\tan\phi,$$
$$\dot{p} = (c_1 r + c_2 p)\cdot r\tan\phi + c_3 L_{\zeta_L,\zeta_V,\zeta_R,\beta} + c_4 N_{\zeta_L,\zeta_V,\zeta_R,\beta},$$
$$\dot{r} = (c_8 p - c_2 r)\cdot r\tan\phi + c_4 L_{\zeta_L,\zeta_V,\zeta_R,\beta} + c_9 N_{\zeta_L,\zeta_V,\zeta_R,\beta}.$$

where p is the rolling moment, r is the yawing moment, and $Y_{\zeta_L,\zeta_V,\zeta_R,\beta}$, $L_{\zeta_L,\zeta_V,\zeta_R,\beta}$, and $N_{\zeta_L,\zeta_V,\zeta_R,\beta}$ are (linear) functions of the control angles $(\zeta_L, \zeta_V, \zeta_R)$ and β.

The physical environment E_{Main} clearly depends on the subcontrollers's physical environments. Each control angle ζ_M in E_{Main} must be the same as the corresponding surface angle α_M, but the ODEs of the subcontrollers cannot be directly "plugged" into E_{Main}, because the main controller and the subcontrollers have *different periods with local clock skews*. The continuous connections between the physical environments are specified by the time-invariant constraint: $\forall t.\,(\zeta_L(t) = \alpha_L(t)) \wedge (\zeta_V(t) = \alpha_V(t)) \wedge (\zeta_R(t) = \alpha_R(t))$.

[4]For example, the new value of $rate_M$ can be given by $\text{sign}(goal_M - v_M)\cdot\min(\text{abs}(goal_M - v_M)/T, max_M)$.

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° and the goal direction from the pilot is fixed (e.g., 30°). In the analysis, we assume the sampling time $t_I = 0$ 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 $\delta = 0.001$, 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.

6.2 Networked Water Tank Controllers

In this benchmark, adapted from [16], a number of water tanks are connected by pipes as shown in Fig. 9. The water level in each tank is controlled by a pump in the tank, and depends on the pump's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the water level of the input tank. The water level x_i of tank i changes according to the nonlinear ODEs:

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

We set $x_0 = 0$ for the leftmost tank 1. Each pipe controller performs its transitions according to its local clock and sets the pump to `on` if $x_i \le L_m$ and to `off` if $x_i > L_M$. The desired safety property is that each water level x_i is in the range $I = [L_m - \eta, L_M + \eta]$, expressed as $(\forall t)\, x_i(t) \in I$.

We have verified the safety property for *any number* of connected water tanks for unbounded time *with respect to clock skews* using compositional inductive reasoning. First, assuming that the input water level x_{i-1} is in I, we show that x_i is in a tighter range $I' = [L_m - \eta', L_M + \eta']$ with $\eta' < \eta$ during one round $[0, T]$.[6] Next, assuming that the input level x_{i-1} is in I, we show that $(\forall t)\, x_i(t) \in I$ is an inductive condition for one round (that is, $x_i(0) \in I$, $\phi_{\mathcal{E}_T \restriction \Pi E}$, and $\forall t \in [0,T].\, c_{in}^{i-1}(x_{i-1}(t))$ implies $\forall t \in [0,T].\, x_i(t) \in I$).

[5]E.g., the output condition for the left subcontroller is $c_{out}^L(v_L, \alpha_L(t)) \equiv \forall t \in [0,T].\,\text{abs}(\alpha_L(t) - v_L) < \gamma$.

[6]I/O conditions are $c_{in}^i(x_{i-1}(t)) \equiv (\forall t \in [0,T].\, x_{i-1}(t) \in I)$ and $c_{out}^i(x_i(t)) \equiv (\forall t \in [0,T].\, x_i(t) \in I')$.

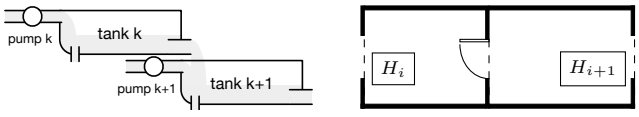

Figure 9: Connected water tanks, and rooms.

We have proved this compositional safety property for maximal clock skew $\epsilon = 30\,\text{ms}$, sampling time $t_I = 20\,\text{ms}$, and response time $t_R = 100\,\text{ms}$, with precision $\delta = 0.001$ using dReal (the analysis took 4.3 seconds). However, if $\epsilon = 150\,\text{ms}$, 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 $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the temperatures of the connected rooms, and changes according to the ODEs:

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

In each transition, a controller of room i turns on the heater if $x_i \le T_m$, and turns it off if $x_i > T_M$. The safety property is that the temperature 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 $\epsilon = 2\,\text{ms}$, sampling time $t_I = 10\,\text{ms}$, and response time $t_R = 200\,\text{ms}$, using dReal with precision $\delta = 0.001$ (the analysis took 2.6 seconds). However, if $\epsilon = 20\,\text{ms}$, 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 actuating.

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

[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 *IEEE RTSS*, 2009.

[2] 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. *Sci. Comp. Prog.*, 103, 2015.

[3] K. Bae, J. Meseguer, and P. C. Ölveczky. Formal patterns for multirate distributed real-time systems. *Sci. Comp. Program.*, 91:3–44, 2014.

[4] K. Bae, P. Ölveczky, S. Kong, and S. Gao. SMT-based analysis of virtually synchronous hybrid systems. http://kquine.github.io/vsdh/techrep.pdf.

[5] K. Bae and P. C. Ölveczky. Hybrid Multirate PALS. In *Logic, Rewriting, and Concurrency*, volume 9200 of *LNCS*. Springer, 2015.

[6] S. Bogomolov, C. Herrera, M. Muñiz, B. Westphal, and A. Podelski. Quasi-dependent variables in hybrid automata. In *HSCC*. ACM, 2014.

[7] R. P. Collinson. *Introduction to avionics systems*. Springer, 2013.

[8] S. Gao, J. Avigad, and E. M. Clarke. δ-complete decision procedures for satisfiability over the reals. In *IJCAR*, volume 7364 of *LNCS*. Springer, 2012.

[9] S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *CADE*, volume 7898 of *LNCS*. Springer, 2013.

[10] S. Gao, S. Kong, and E. M. Clarke. Satisfiability modulo ODEs. In *FMCAD*. IEEE, 2013.

[11] M. Hendriks, G. Behrmann, K. G. Larsen, P. Niebert, and F. W. Vaandrager. Adding symmetry reduction to Uppaal. In *FORMATS*, volume 2791 of *LNCS*, 2003.

[12] T. T. Johnson and S. Mitra. A small model theorem for rectangular hybrid automata networks. In *FMOODS/FORTE*. LNCS 7273, Springer, 2012.

[13] T. T. Johnson and S. Mitra. Anonymized reachability of hybrid automata networks. In *FORMATS*, volume 8711 of *LNCS*. Springer, 2014.

[14] N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Infor. and Comput.*, 185(1), 2003.

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

[16] J. Raisch, E. Klein, S. O'Young, C. Meder, and A. Itigin. Approximating automata and discrete control for continuous systems. In *Hybrid Systems V*, volume 1567 of *LNCS*. Springer, 1999.