SMT-Based Analysis of Virtually Synchronous Distributed Hybrid Systems

Kyungmin Bae[1,2], Soonho Kong[1], Sicun Gao[3], Peter Csaba Ölveczky[4], and Edmund M. Clarke[1]

[1] Carnegie Mellon University
[2] SRI International
[3] Massachusetts Institute of Technology
[4] University of Oslo

Abstract. We present verification techniques for virtually synchronous distributed control systems with *interconnected* physical environments. Such cyber-physical systems are notoriously hard to verify, due to their combination of nontrivial continuous dynamics, asynchronous communication, network delays, imprecise local clocks, etc. To simplify their analysis, we first extend the PALS methodology—which 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 explain how a number of verification problems for synchronous Hybrid PALS models, such as bounded reachability, inductive analysis, and compositional reasoning, can be reduced to SMT solving over nonlinear theories of the real numbers. We illustrate the Hybrid PALS verification methodology on a number of virtually synchronous distributed hybrid systems, including a multirate control system for turning an airplane.

1 Introduction

Virtually synchronous distributed hybrid systems consist of a number of *distributed* controllers—where each controller may interact with its local physical environment having continuous dynamics which, furthermore, can be correlated—that should logically behave in a synchronous way. This important class of cyber-physical systems (CPSs) includes avionics, automotive, robotics, and medical systems. The devices may operate at different frequencies, but communication happens at their hyper-period boundary. Designing and analyzing such systems is difficult because of the interrelated continuous behaviors combined with clock skews, network delays, execution times, and so on. In particular, it is very hard to apply modeling and analysis techniques based on hybrid automata since: (i) such systems explicitly take into account asynchronous communication with real-time constraints; and (ii) continuous interactions between distributed components can make the system's behavior *nonlinear*.

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 [1,3,17] 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 therefore introduce and develop *Hybrid PALS* for virtually synchronous distributed hybrid systems. Hybrid PALS significantly extends and modifies the approach in [4] to achieve a bisimulation equivalence between Hybrid PALS synchronous models and their asynchronous counterparts.

This equivalence means that verifying a virtually synchronous multirate distributed hybrid system reduces to verifying its underlying synchronous model. Hybrid PALS allows us to abstract from network delays, message buffering, asynchronous communication, etc. However, the times at which physical states are sampled or actuator commands are sent to the environment cannot be abstracted from. Since these events are triggered by imprecise local clocks, we also need to take into account those clocks. Furthermore, the physical environments of virtually synchronous components are often *tightly coupled*, so 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 precisely analyze synchronous Hybrid PALS models.

This paper therefore presents SMT solving techniques to address the challenges of analyzing synchronous 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 $\delta > 0$ [8,10].[5] The discrete parts and the continuous parts are encoded separately, so that ODE solving is considered *only after* the discrete parts of the system have been fully analyzed. We show that standard verification problems for hybrid systems are encoded as SMT formulas. In addition to bounded reachability, for scalability reasons, we describe how unbounded time inductive reasoning and compositional assume-guarantee reasoning can be encoded in SMT for synchronous Hybrid PALS models.

We have applied our techniques on a range of non-trivial nonlinear systems, including: an 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. Owing to the bisimulation equivalence, we use simple synchronous models to analyze such complex distributed hybrid models.

[5] The number δ, provided by the user, is the bound on numerical errors that is tolerable in the analysis.

To summarize, the new contributions in this paper (also compared to [4]) 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 small toy examples in [4]); 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 6 discusses related work. Section 2 gives a background on PALS. Section 3 introduces Hybrid PALS. Section 4 shows SMT encodings for Hybrid PALS models and their analysis. Section 5 gives an overview of the Hybrid PALS verification case studies. Finally, Section 7 gives some concluding remarks.

2 Preliminaries on PALS

PALS transforms a *synchronous design SD* into a distributed real-time system $\mathcal{MA}(SD, T, \Gamma)$ with period T, satisfying the same temporal logic properties, provided that the underlying infrastructure guarantees bounds Γ on network delays, execution times, and clock skews. This section overviews the synchronous models SD, the distributed models $\mathcal{MA}(SD, T, \Gamma)$, and the relationship between SD and $\mathcal{MA}(SD, T, \Gamma)$ (we refer to [3, 17] for details).

2.1 Discrete Synchronous Models.

The synchronous model SD is specified as an *ensemble* $\mathcal{E}$ of (nondeterministic) state machines with input and output ports. In each iteration, a state 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 *is a tuple* $M = (D_i, S, D_o, \delta_M)$*, where (i)* $D_i = D_{i_1} \times \cdots \times D_{i_n}$ *an input set (a value to the* k*-th* input port *is an element of* D_{i_k}*), (ii)* S *a set of states, (iii)* $D_o = D_{o_1} \times \cdots \times D_{o_m}$ *an output set, (a value from the* j*-th* output port *is an element of* D_{o_j}*) and (iv)* $\delta_M \subseteq (D_i \times S) \times (S \times D_o)$ *a total transition relation.*

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

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

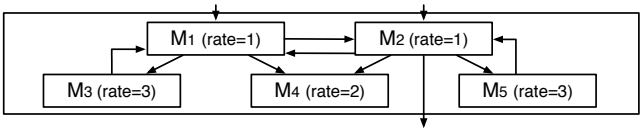


Fig. 1. A multirate ensemble $\mathcal{E}$, with M_1 and M_2 slow machines.

Definition 2. *A tuple* $\mathcal{E} = (J_S, J_F, e, \{M_j\}_{j\in J_s\cup J_F}, E, src, rate, adap)$ *denotes a* multirate ensemble*, where:*

- J_S *is a set of "slow machine" indices;*
- J_F *is a set of "fast machine" indices* ($J_S \cap J_F = \emptyset$)*;*
- $e \notin J_S \cup J_F$ *is the "ensemble interface" index;*
- $\{M_j\}_{j\in J_S\cup J_F}$ *is a family of typed machines;*
- $E = (D_i^e, D_o^e)$ *is the ensemble's* interface *with* D_i^e *the* input set *and* D_o^e *its* output set*;*
- *src is a wiring diagram assigning to each input port* (j, n) *(input port* n *of machine* j*) its "source" output port;*
- *rate assigns to each fast machine its rate; and*
- *adap assigns an input adaptor to each machine.*

To formally define a synchronous composition, we first recall some notations. The k-step deceleration of a machine M performs k transitions in one step, where each input and output port handles a k-tuple of values. The adaptor closure of M with an *input adaptor* $\alpha = \{\alpha_j : D'_j \to D_{i_j}\}_{j\in\{1,\dots,n\}}$ is the machine in which each input port j is enclosed by each input adaptor function α_j.

Definition 3. *For a machine* $M = (D_{i_1} \times \cdots \times D_{i_n}, S, D_{o_1} \times \cdots \times D_{o_m}, \delta_M)$*, its* k*-step* deceleration *is* $M^{\times k} = (D_{i_1}^k \times \cdots \times D_{i_n}^k,\ S,\ D_{o_1}^k \times \cdots \times D_{o_m}^k,\ \delta_{M^{\times k}})$*, where for* $\pi_j(\boldsymbol{d})$ *denoting the* j*-th component of a* k*-tuple* $\boldsymbol{d}$*:*

$$(((\boldsymbol{i}_1, \dots, \boldsymbol{i}_n), s_0), (s_k, (\boldsymbol{o}_1, \dots, \boldsymbol{o}_m))) \in \delta_{M^{\times k}}$$

iff $\exists s_1, \dots, s_{k-1} \in S$ *such that*

$$\bigwedge_{j=0}^{k-1} (((\pi_j(\boldsymbol{i}_1), \dots, \pi_j(\boldsymbol{i}_n)), s_j), (s_{j+1}, (\pi_j(\boldsymbol{o}_1), \dots, \pi_j(\boldsymbol{o}_m)))) \in \delta_M.$$

Definition 4. *Given a machine* $M = (D_i, S, D_o, \delta_M)$ *and an input adaptor* $\alpha = \{\alpha_j : D'_j \to D_{i_j}\}_{j\in\{1,\dots,n\}}$*, where* $\alpha(d_1, \dots, d_n) = (\alpha_1(d_1), \dots, \alpha_n(d_n))$*, the* adaptor closure *is the machine* $M_\alpha = ((D'_1 \times \cdots \times D'_n), S, D_o, \delta_{M_\alpha})$ *such that* $((\boldsymbol{i}, s), (s', \boldsymbol{o})) \in \delta_{M_\alpha} \iff ((\alpha(\boldsymbol{i}), s), (s', \boldsymbol{o})) \in \delta_M$*.*

The *synchronous composition* of a multirate ensemble $\mathcal{E}$ is equivalent to a single machine $M_\mathcal{E} = (D_i^\mathcal{E}, S^\mathcal{E}, D_o^\mathcal{E}, \delta_{M_\mathcal{E}})$. If a machine in $\mathcal{E}$ has a feedback wire connected to itself or to another component, then the output becomes an input of

the destination in the next iteration. That is, $M_{\mathcal{E}}$'s states $S^{\mathcal{E}}$ consist of the states S_j of its subcomponents M_j and the "feedback" outputs D^j_{OF} (i.e., the outputs from M_j to some machine in $\mathcal{E}$). For example, the synchronous composition $M_{\mathcal{E}}$ of the ensemble $\mathcal{E}$ in Fig. 1 is the machine given by the outer box.

Definition 5. *For an ensemble* $\mathcal{E} = (J_S, J_F, e, \{M_j\}_{j\in J_s\cup J_F}, E, src, rate, adap)$. *let* $\overline{M}_j = (M_j^{\times rate(j)})_{adap(j)}$ *for* $j \in J_S \cup J_F$. *The* synchronous composition *of* $\mathcal{E}$ *is the machine* $M_{\mathcal{E}} = (D^{\mathcal{E}}_i, S^{\mathcal{E}}, D^{\mathcal{E}}_o, \delta_{\mathcal{E}})$, *where: (i)* $D^{\mathcal{E}}_i = D^e_o$ *and* $D^{\mathcal{E}}_o = D^e_i$; *(ii)* $S^{\mathcal{E}} = \prod_{j\in J_S\cup J_F}(S_j \times D^j_{OF})$; *and (iii)* $\delta_{\mathcal{E}} \subseteq (D^{\mathcal{E}}_i \times S^{\mathcal{E}}) \times (S^{\mathcal{E}} \times D^{\mathcal{E}}_o)$ *that "combines" the transitions of* $\{\overline{M}_j\}_{j\in J_S\cup J_F}$ *into a synchronous step:*

$$\begin{aligned} &\big((\boldsymbol{i}, \{s_j, \boldsymbol{f}_j\}_{j\in J_S\cup J_F}), (\{s'_j, fout_l(\boldsymbol{o}'_l)\}_{j\in J_S\cup J_F}, in_e(\{\boldsymbol{o}'_j\}_{j\in J_S\cup J_F}))\big) \in \delta_{\mathcal{E}} \\ \iff\ &\textit{for each } l \in J_S \cup J_F,\ \big((in_l(\boldsymbol{i}, \{\boldsymbol{f}_j\}_{j\in J_S\cup J_F}), s_l), (s'_l, \boldsymbol{o}'_l)\big) \in \delta_{\overline{M}_l}. \end{aligned}$$

where $fout_j(\boldsymbol{o}_j)$ *is the feedback part of* $\overline{M}_j$*'s output* $\boldsymbol{o}_j$, $in_e(\{\boldsymbol{o}'_j\}_{j\in J_S\cup J_F})$ *is the output to the interface* e *from outputs* $\{\boldsymbol{o}'_j\}_{j\in J_S\cup J_F}$ *of the subcomponents, and* $in_l(\boldsymbol{i}, \{\boldsymbol{f}_j\}_{j\in J_S\cup J_F})$ *is the input to* $\overline{M}_l$ *from interface input* $\boldsymbol{i}$ *and feedback outputs* $\{\boldsymbol{f}_j\}_{j\in J_S\cup J_F}$, *all of which are given by the wiring diagram* src.

The transition system for $M_{\mathcal{E}}$ is a tuple $ts(M_{\mathcal{E}}) = (S^{\mathcal{E}} \times D^{\mathcal{E}}_i, \longrightarrow_{\mathcal{E}})$, where $(\boldsymbol{s}_1, \boldsymbol{i}_1) \longrightarrow_{\mathcal{E}} (\boldsymbol{s}_2, \boldsymbol{i}_2)$ iff an ensemble in state $\boldsymbol{s}_1$ with input $\boldsymbol{i}_1$ from the interface has a transition to state $\boldsymbol{s}_2$ (i.e., $\exists \boldsymbol{o}\ ((\boldsymbol{i}_1, \boldsymbol{s}_1), (\boldsymbol{s}_2, \boldsymbol{o})) \in \delta_{M_{\mathcal{E}}}$). Notice that this model is an untimed model (apart from the period T): the result of applying a transition is independent of *when* the transition is applied in a round.

2.2 PALS Distributed Real-Time Models

PALS assumes that the underlying infrastructure provides performance bounds $\Gamma = (\epsilon, \alpha_{\min}, \alpha_{\max}, \mu_{\min}, \mu_{\max})$, with (i) ϵ a maximal clock skew with respect to the global clock, (ii) $[\alpha_{\min}, \alpha_{\max}]$ bounds for executing a transition, and (iii) $[\mu_{\min}, \mu_{\max}]$ bounds for the network transmission delay.

Each component in the distributed model $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ is composed of a machine in $\mathcal{E}$ and *wrappers* around it, as illustrated in Fig. 2. The outermost wrapper is the PALS wrapper, which encloses an input adaptor wrapper, which encloses either a (slow) machine or a k-machine wrapper, which encloses a (fast) machine. Each machine in $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ performs at its own rate according to its local clock. At the beginning of its periods, it reads its input from the layer above, performs a transition, and then generates the outputs.

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

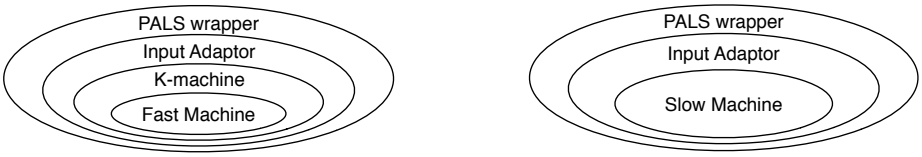


Fig. 2. The wrapper hierarchies in PALS distributed real-time models.

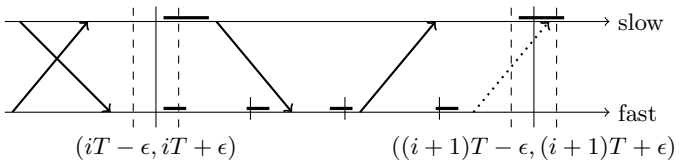


Fig. 3. Timeline for $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ (with $k = 4$ and $k' = 3$). Diagonal arrows denote network transmission and short horizontal lines denote the execution. The dotted arrow illustrates a deadline miss caused when the fast machine finishes all its transitions.

buffer are sent out into the network. All inputs are read in a round-consistent way if $T \geq 2\epsilon + \mu_{max} + \max(2\epsilon - \mu_{min}, \alpha_{max})$ [17].[6]

An input adaptor wrapper reads the inputs from the PALS wrapper and applies input adaptor functions for each global period T according to its local clock, in order to get either a single value from a k-tuple input for slow machines, or a k-tuple of values from a single input for fast machines. A k-machine wrapper (i) extracts each value from the k-tuple input and delivers it to the enclosed fast machine at each fast period T/k, and (ii) delivers the *k-tuples* from the outputs of the fast machine to its outer layer at each global period T.

As depicted in Fig. 3, a fast machine M_f may *not* be able to finish all of its k internal transitions in a global round *before* the outputs must be sent to arrive before the next round. The number of transitions that M_f can perform before the deadline is $k' = 1 + \lfloor \max(T - (2\epsilon + \mu_{max} + \alpha_{max_f}), 0) \cdot (k/T) \rfloor$, where $\alpha_{\max_f}$ is the maximal execution time for M_f. If $k' < k$, then M_f's k-machine wrapper only sends the first k' values (followed by $k - k'$ "don't care" values $\perp$). The input adaptor of each input port whose source is M_f must be $(k'+1)$-*oblivious*, i.e., it ignore the last $k - k'$ values $v_{k'+1}, \ldots, v_k$ in a k-tuple $(v_1, \ldots, v_k)$.

2.3 Relating the Synchronous and Distributed Models

Stable state of the distributed model $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ 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 [17]. As depicted in Fig. 3, network transmission in

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

$\mathcal{MA}(\mathcal{E}, T, \Gamma)$ can happen only in the time interval $(iT+\epsilon, (i+1)T-\epsilon)$. In stable states at times $iT-\epsilon$, all the input buffers of the PALS wrappers are full, and all the other input and output buffers are empty. Therefore, there exists the function $sync : Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma)) \to S^{\mathcal{E}} \times D_i^{\mathcal{E}}$ that maps stable states to the corresponding states of the synchronous composition $M_{\mathcal{E}}$ in a natural way.

Definition 6. *Given a stable state $C \in Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma))$, $sync(C)$ is a pair $(\{s_j, \boldsymbol{f}_j\}_{j \in J_S \cup J_F}, \boldsymbol{i}) \in S^{\mathcal{E}} \times D_i^{\mathcal{E}}$ such that: (i) the states of the typed machines in C give the states $\{s_j\}_{j \in J_S \cup J_F}$ of the machines; and (ii) the values in the input buffers of the PALS wrappers in C give the values $\{\boldsymbol{f}_j\}_{j \in J_S \cup J_F}$ in the feedback wires and the input $\boldsymbol{i}$ from the ensemble interface.*

Assuming $(k'+1)$-oblivious of the input adaptors, we can relate two stable states by $C_1 \sim_{obi} C_2$ iff their machine states are identical and their corresponding input buffer contents *cannot* be distinguished by input adaptors.

Definition 7. *For two stable states $C_1, C_2 \in Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma))$, $C_1 \sim_{obi} C_2$ iff (i) each machine M_j has the same machine state in C_1 and C_2; and (ii) for each machine M_j, if $k' \leq rate(j)$ is its cutoff number, then each input buffer of M_j's PALS wrapper has the same first k' values in C_1 and C_2.*

Big-step transitions $\longrightarrow_{st}$ are defined between two stable states, and those in the transition system $ts(Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma))) = (Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma)), \longrightarrow_{st})$ are related to single synchronous steps of the transition system $ts(M_{\mathcal{E}})$.

Theorem 1. *[3] The binary relation $(\sim_{obi}; sync)$ is a* bisimulation *between the transition systems $ts(M_{\mathcal{E}})$ and $ts(Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma)))$.*

Also, if the state labeling function $\mathcal{L} : S^{\mathcal{E}} \times D_i^{\mathcal{E}} \to 2^{AP}$ for an ensemble $\mathcal{E}$ and it cannot distinguish between $\sim_{obi}$-equivalent states, then the transition systems $ts(M_{\mathcal{E}})$ and $ts(Stable(\mathcal{MA}(\mathcal{E}, T, \Gamma)))$ (more precisely, their corresponding Kripke structures) satisfy the same CTL^* formulas [3].

3 Hybrid PALS

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

In Hybrid PALS, the standard PALS models $\mathcal{E}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ are both nondeterministic models defined for *any possible* environment behaviors. The *environment restrictions* $\mathcal{MA}(\mathcal{E}, T, \Gamma) \upharpoonright E$ and $\mathcal{E} \upharpoonright E$ define the behavior of the

models constrained by the physical environment E. To have any control over when values are read from, and sent to, physical environments, *sampling and response timing policies* are added to the Hybrid PALS models. We then prove a bisimulation equivalence relating $\mathcal{E} \upharpoonright E$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \upharpoonright E$.

3.1 Controlled Physical Environments

A state of a physical environment is a tuple $\boldsymbol{v} = (v_1, \ldots, v_l) \in \mathbb{R}^l$ of its physical parameters $\boldsymbol{x} = (x_1, \ldots, x_l)$, and the behavior of $\boldsymbol{x}$ can be modeled by ODEs that specify *trajectories* $\tau_1, \ldots, \tau_l$ of the parameters $\boldsymbol{x}$ over time. A trajectory of duration T is a function $\tau : [0, T] \to \mathbb{R}$. Let $\mathcal{T}$ denote the set of all trajectories, and $\boldsymbol{\tau}(t) = (\tau_1(t), \ldots, \tau_l(t))$ for an l-tuple of trajectories $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_l)$.

Definition 8. *Given a trajectory $\tau : [0, T] \to \mathbb{R}$ and $u \in [0, T]$, τ's* prefix *is denoted by $\tau \trianglelefteq u$ (that is, $\tau \trianglelefteq u(t) = \tau(t)$ for $t \in [0, u]$), and τ's* suffix *is denoted by $\tau \trianglerighteq u$ (that is, $\tau \trianglerighteq u(t) = \tau(t + u)$ for $t \in [0, T - u]$).*

A physical environment E_M of machine M is specified as a *controlled physical environment* that defines every possible trajectory of its physical parameters $\boldsymbol{x}$ for the control commands from M. For a state $\boldsymbol{v} \in \mathbb{R}^l$, a control command a, and a duration $t \in \mathbb{R}$, a physical environment E_M gives a trajectory $\boldsymbol{\tau} \in \mathcal{T}^l$ of its parameters $\boldsymbol{x}$ of duration t, as illustrated in Fig. 4 (e.g., state v_1, command a, duration $t_2 - t_1$ yields trajectory τ_1).

Definition 9. *For $l \in \mathbb{N}$, an l-dimensional* controlled physical environment *is a tuple $E_M = (C, \boldsymbol{x}, \Lambda)$, where:*

- *C is a set of* control commands *from the controller M;*
- *$\boldsymbol{x} = (x_1, \ldots, x_l)$ is a vector of real number variables; and*
- *$\Lambda \subseteq (C \times \mathbb{R}^l \times \mathbb{R}_{\geq 0}) \times \mathcal{T}^l$ is a* physical transition relation*: $((a, \boldsymbol{v}, t), \boldsymbol{\tau}) \in \Lambda$ iff for a control command $a \in C$ that lasts for duration t, E_M's physical state $\boldsymbol{x}$ follows the trajectory $\boldsymbol{\tau} \in \mathcal{T}^l$ from state $\boldsymbol{\tau}(0) = \boldsymbol{v} \in \mathbb{R}^l$.*[7]

Several physical environments may 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 another environment E_{M_2}, then the time-invariant constraint is the formula $\forall t.\, x_1(t) = x_2(t)$ with variable t for time. Without loss of generality, we assume that parameter names of different components are all different.

Example 1. Consider a simple distributed CPS controller to roll an airplane by moving its *ailerons* (a hinged surface attached to the end of the left or the right wing), illustrated in Fig. 5. Each aileron subcontroller moves the corresponding aileron towards the goal angle given by the main controller. The subcontrollers

[7] If E_M defines a trajectory τ, then E_M should also define every prefix of τ. That is, we also require that if $((a, \boldsymbol{v}, t), \boldsymbol{\tau}) \in \Lambda$, then $((a, \boldsymbol{v}, t'), \boldsymbol{\tau} \trianglelefteq t') \in \Lambda$ for any $t' < t$.

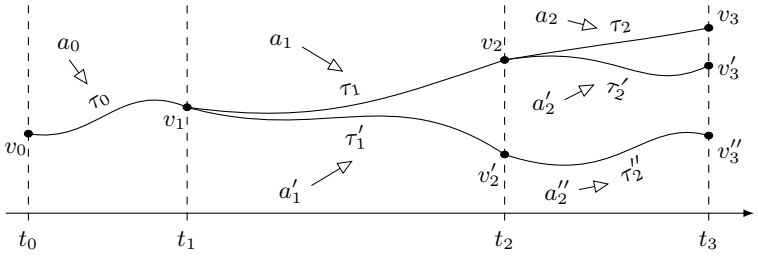


Fig. 4. A controlled physical environment E_M; for example, $((a_0, v_0, t_1 - t_0), \tau_0) \in \Lambda$, $((a_1, v_1, t_2 - t_1), \tau_1) \in \Lambda$, $((a_1', v_1, t_2 - t_1), \tau_1') \in \Lambda$, etc.

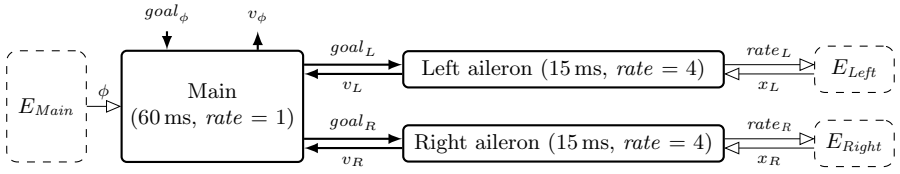


Fig. 5. The simple CPS controller to roll an airplane. The period of the main controller (rate 1) is four times longer than the periods of the subcontrollers (rate 4).

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 aileron angle x_M for subcontroller M changes according to the moving rate $rate_M$ (the control command from M). The continuous dynamics of x_M is specified by the ODE $\dot{x}_M = rate_M$. The controlled physical environment of M is given by $E_M = (\mathbb{R}, x_M, \Lambda_M)$, where: (i) $\mathbb{R}$ is the domain of the control command $rate_M$; (ii) x_M is the physical parameter for the aileron angle; and (iii) $\Lambda_M \subseteq (\mathbb{R} \times \mathbb{R} \times \mathbb{R}_{\geq 0}) \times \mathcal{T}$ is the physical transition relation such that

$$\big((rate_M, v_M, 15\,\text{ms}), \tau\big) \in \Lambda_M \iff \forall t \in [0, 15\,\text{ms}].\ \tau(t) = v_M + \int_0^t rate_M \,\mathrm{d}t.$$

Disregarding the yawing effect caused by the rolling, the physical dynamics of an aircraft is specified as the ODEs $\dot{\phi} = p$ and $\dot{p} = c(\delta_R - \delta_L)$, where ϕ is the roll angle, p is the rolling moment, δ_R is the angle of the right aileron, δ_L is the angle of the left aileron, and c is a constant determined by the geometry of the aircraft. The controlled physical environment of the main controller is given by $E_{Main} = (\{*\}, (\phi, p, \delta_L, \delta_R), \Lambda_{Main})$, where: (i) $\{*\}$ is the singleton set to indicate that E_{Main} has no control command; (ii) $(\phi, p, \delta_L, \delta_R)$ are the physical parameters of E_{Main}; and (iii) $\Lambda_{Main} \subseteq (\{*\} \times \mathbb{R}^4 \times \mathbb{R}_{\geq 0}) \times \mathcal{T}^4$ is the physical

transition relation such that

$$\begin{aligned}&\big((*, (v_\phi, v_p, v_{\delta_L}, v_{\delta_R}), 60\,\text{ms}), (\tau_\phi, \tau_p, \tau_{\delta_L}, \tau_{\delta_R})\big) \in \Lambda_{Main}\\ \iff &\forall t \in [0, 60\,\text{ms}].\ \begin{bmatrix}\tau_\phi\\ \tau_p\end{bmatrix}(t) = \begin{bmatrix}v_\phi\\ v_p\end{bmatrix} + \int_0^t \begin{bmatrix}p(t)\\ c(\tau_{\delta_R}(t) - \tau_{\delta_L}(t))\end{bmatrix} \mathrm{d}t.\end{aligned}$$

The behavior of E_{Main} depends on the trajectories of the control angles (δ_L, δ_R), determined by the subcontrollers's physical environments. The control angles δ_L and δ_R must be the same as the respective aileron angles x_L and x_R of the subcontrollers. However, because the main controller and the subcontrollers have *different periods with local clock skews*, the ODEs of the subcontrollers cannot be directly "plugged" into E_{Main}. Instead, their physical correlations can be specified as the time-invariant constraint

$$\forall t.\, (\delta_L(t) = x_L(t)) \wedge (\delta_R(t) = x_R(t)).$$

3.2 Sampling and Response Timing

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

Fig. 6 depicts the behavior of the controller M with respect to its physical environment E_M and a periodic local clock c_M in a PALS distributed model for a time interval $[iT - \epsilon, (i+1)T - \epsilon]$ of duration T:

1. The $(i+1)$-th period begins at time $c_M(i) \in (iT - \epsilon, iT + \epsilon)$. Because of PALS bounds, M has already received all the inputs $\boldsymbol{i}$ before time $iT - \epsilon$.
2. M samples the appropriate values of its environment E_M. The sampling takes time t_I and the physical state $\boldsymbol{v}_I$ of E_M is read at time $c_M(i) + t_I$.
3. M executes its transition based on the inputs $\boldsymbol{i}$, the sampled physical state $\boldsymbol{v}_I$, and the current machine state s.
4. After the execution, the machine state changes to s', and the new controller command a is sent to E_M at time $c_M(i) + t_R$, where t_R is the response time for the transition execution and the actuator processing.
5. The new outputs $\boldsymbol{o}$ 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 its physical environment E_M only depends on M's current state s. In Fig. 6, a current control command a remains effective until the execution of M ends at time $u_R = c_M(i) + t_R$, and then a new control command a' takes effect. That is, E_M defines trajectories $\boldsymbol{\tau}$ in an interval $[iT - \epsilon, (i+1)T - \epsilon]$ of duration T with respect to (a, a', u_R).

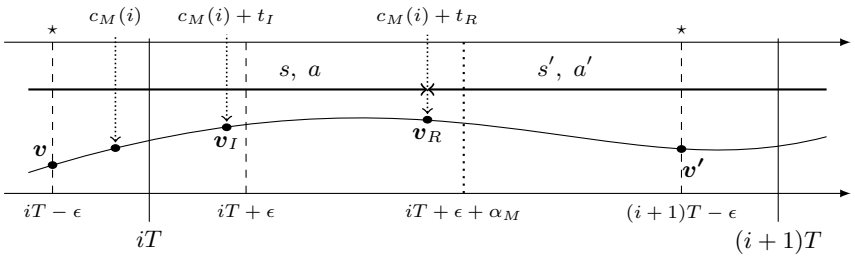


Fig. 6. Timeline for an environment-restricted controller with a local clock c_M, an environment sampling time t_I, and an response time t_R.

Definition 10. *For a physical environment $E_M = (C, \boldsymbol{x}, \Lambda)$, trajectories $\boldsymbol{\tau} \in \mathcal{T}^l$ of duration T are* realizable *with respect to control commands $a, a' \in C$ and a response duration $u_R \in \mathbb{R}$, denoted by $\boldsymbol{\tau} \in \mathcal{R}^T_{E_M}(a, u_R, a')$, iff:*

$$\big((a, \boldsymbol{\tau}(0), u_R), \boldsymbol{\tau} \trianglelefteq u_R\big) \in \Lambda \ \ \wedge \ \ \big((a', \boldsymbol{\tau}(u_R), T - u_R), \boldsymbol{\tau} \trianglerighteq u_R\big) \in \Lambda.$$

The sampling times of a controller M depends on its state s, and the response times of M depend on its state s and input $\boldsymbol{i}$. In order to compose M with its physical environment E_M, we define the "interface" between the controller M and its environment E_M, given by the following projection functions:

Definition 11. *The* interface *of machine $M = (D_i, S, D_o, \delta_M)$ is given by the projection functions $\pi = (\pi_T, \pi_R, \pi_I, \pi_C)$, for $s \in S$ and $\boldsymbol{i} \in D_i$:*

- *$\pi_T(s) \in \mathbb{N}$ a round number (i.e., state s is in the $(\pi_T(s)+1)$-th iteration);*
- *$\pi_R(s, \boldsymbol{i}) \in \mathbb{R}$ an environment response time;*
- *$\pi_I(s) \in \mathbb{R}$ an environment sampling time, where $\pi_I(s) \leq \pi_R(s, \boldsymbol{i})$; and*
- *$\pi_C(s) \in C$ the control command of M to E_M.*

3.3 Environment-Restricted Controllers

A controller M is a *nondeterministic* machine parameterized by any behavior of its physical environment E_M. Such a controller M needs to observe "some" physical parameters of E_M (not necessary all parameters of E_M). Therefore, M has a state space of the form $S \times \mathbb{R}^m$, where $\mathbb{R}^m$ is the state space of the m physical parameters that M 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 its state space $\mathbb{R}^l$. For state $\boldsymbol{v} \in \mathbb{R}^l$ of E_M, let $\pi_O(\boldsymbol{v}) \in \mathbb{R}^m$ denote the observable part of $\boldsymbol{v}$.

The *environment restriction* $M \restriction E_M$ is then defined as a normal typed machine with a combined state space $S \times \mathbb{R}^l$, which records "snapshots" of E_M's physical states at times $iT - \epsilon$. A transition of $M \restriction E_M$ from state $(s, \boldsymbol{v})$ to $(s', \boldsymbol{v'})$ corresponds to realizable trajectories $\boldsymbol{\tau}$ for an interval $[iT - \epsilon, (i+1)T - \epsilon]$ with respect to the control commands $\pi_C(s)$ 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 $\pi_O(\boldsymbol{\tau}(u_I))$ of E_M sampled after the sampling duration $u_I = (c_M(i) + \pi_I(s)) - (iT - \epsilon)$.

Definition 12. *Given a controller $M = (D_i, S \times \mathbb{R}^m, D_o, \delta_M)$, an l-dimensional physical environment $E_M = (C, \boldsymbol{x}, \Lambda)$, an interface π, and a period $T \in \mathbb{R}$, the* environment restriction *of M by E_M is the typed machine*

$$M \upharpoonright_\pi E_M = (D_i, S \times \mathbb{R}^l, D_o, \delta_{M \upharpoonright_\pi E_M}),$$

where $((\boldsymbol{i}, (s, \boldsymbol{v})), ((s', \boldsymbol{v'}), \boldsymbol{o})) \in \delta_{M \upharpoonright_\pi E_M}$ iff for the round number $i = \pi_T(s)$, $u_I = (c_M(i) + \pi_I(s)) - (iT - \epsilon)$, and $u_R = (c_M(i) + \pi_R(s)) - (iT - \epsilon)$:

- *$\boldsymbol{\tau}(0) = \boldsymbol{v}$ and $\boldsymbol{\tau}(T) = \boldsymbol{v'}$ for some $\boldsymbol{\tau} \in \mathcal{R}^T_{E_M}(\pi_C(s), u_R, \pi_C(s'))$.*
- *$\big((\boldsymbol{i}, (s, \pi_O(\boldsymbol{\tau}(u_I)))), ((s', \pi_O(\boldsymbol{v'})), \boldsymbol{o})\big) \in \delta_M$, and $\pi_T(s') = i + 1$.*

Example 2. Consider a subcontroller M to move an aileron in Example 1. The period of a subcontroller M is 15 ms, and the $(i+1)$-th round of M begins at time $c_M(i) \in (i \cdot 15\,\text{ms} - \epsilon, i \cdot 15\,\text{ms} + \epsilon)$ with the maximal clock skew $\epsilon > 0$. In each round, M receives a goal angle g_M, sets a new moving rate rate'_M based on g_M and the observed angle v_M, and sends back v_M. Such a subcontroller M can be specified as the typed machine $M = (\mathbb{R}, \mathbb{N} \times \mathbb{R}^2, \mathbb{R}, \delta_M)$, where

$$\begin{aligned} &\big((g_M, (i, \mathit{rate}_M, v_M)), ((i', \mathit{rate}'_M, v'_M), o)\big) \in \delta_M \\ \iff\ & \mathit{rate}'_M = (g_M - v_M)/15\,\text{ms} \ \wedge\ o = v_m \ \wedge\ i' = i + 1. \end{aligned}$$

The interface π_M of M can be defined by projection functions as follows: (i) $\pi_C(i, \mathit{rate}_M, v_M) = \mathit{rate}_M$ (the controller command); (ii) $\pi_T(i, \mathit{rate}_M, v_M) = i$ (the round number); (iii) $\pi_I(i, \mathit{rate}_M, v_M) = 1\,\text{ms}$ (the environment sampling time); and (iv) $\pi_R((i, \mathit{rate}_M, v_M), g_M) = 2\,\text{ms}$ (the environment response time). For the environment $E_M = (\mathbb{R}, x_M, \Lambda_M)$ in Example 1, its observable state is given by the function $\pi_O(x_M) = x_M$, since M observes the aileron angle x_M.

The environment restriction is defined as a typed machine $M \upharpoonright_{\pi_M} E_M$. Its transitions are associated with the interval $[i \cdot 15\,\text{ms} - \epsilon, (i+1) \cdot 15\,\text{ms} - \epsilon]$. The angle x_M follows a "realizable" trajectory, by the current moving rate rate_M up to the response time $c_M(i) + 2\,\text{ms}$ and then by the new rate rate'_M. The controller M performs a transition using the angle observed at the sampling time $c_M(i) + 1\,\text{ms}$. That is, $M \upharpoonright_{\pi_M} E_M = (\mathbb{R}, \mathbb{N} \times \mathbb{R}^2, \mathbb{R}, \delta_{M \upharpoonright_{\pi_M} E_M})$, where for $u_I = (c_M(i) + 1\,\text{ms}) - (i \cdot 15\,\text{ms} - \epsilon)$ and $u_R = (c_M(i) + 2\,\text{ms}) - (i \cdot 15\,\text{ms} - \epsilon)$:

$$\begin{aligned} &((g_M, (i, \mathit{rate}_M, v_M)), ((i', \mathit{rate}'_M, v'_M), v_M)) \in \delta_{M \upharpoonright_{\pi_M} E_M} \\ \iff\ & \exists \tau \in \mathcal{R}^{15\,\text{ms}}_{E_M}(\mathit{rate}_M, u_R, \mathit{rate}'_M).\ v_M = \tau(0) \ \wedge\ v'_M = \tau(15\,\text{ms}) \wedge \\ & \qquad ((g_M, (i, \mathit{rate}_M, \tau(u_I))), ((i', \mathit{rate}'_M, v'_M), v_M)) \in \delta_M. \end{aligned}$$

Example 3. Now consider the main controller M_{Main} in Example 1. Its period is 60 ms, and its $(i+1)$-th round begins at $c_{Main}(i) \in (i \cdot 60\,\text{ms} - \epsilon, i \cdot 60\,\text{ms} + \epsilon)$. In each round, 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). The main controller can be specified as $M_{Main} = (\mathbb{R}^3, \mathbb{N} \times \mathbb{R}^2, \mathbb{R}^3, \delta_{Main})$, where

$$\begin{aligned} & \big(((g_\phi, v_L, v_R), (i, v_\phi)), ((i', v'_\phi), (o, g_L, g_R))\big) \in \delta_{Main} \\ \iff\ & g_R = 0.3 \cdot (g_\phi - v_\phi) \ \ \wedge \ \ g_L = -g_R \ \wedge \ o = v_\phi \ \ \wedge \ \ i' = i + 1. \end{aligned}$$

The interface π_{Main} can be defined as follows (since there is no controller command, the value of the response time is not important): (i) $\pi_C(i, v_\phi) = *$; (ii) $\pi_I(i, v_\phi) = 3\,\text{ms}$; (iii) $\pi_T(i, v_\phi) = i$; and (iv) $\pi_R((i, v_\phi), (g_\phi, v_L, v_R)) = 10\,\text{ms}$. For $E_{Main} = (\{*\}, (\phi, p, \delta_L, \delta_R), \Lambda_{Main})$ in Example 1, its observable state is given by $\pi_O(\phi, p, \delta_L, \delta_R) = \phi$, since M_{Main} only observes the roll angle ϕ.

The environment restriction of M_{Main} by E_{Main} is then the typed machine $M_{Main} \restriction_{\pi_{Main}} E_{Main} = (\mathbb{R}^3, \mathbb{N} \times \mathbb{R}^4, \mathbb{R}^3, \delta_{M_{Main} \restriction_{\pi_{Main}} E_{Main}})$. Each transition of $M_{Main} \restriction_{\pi_{Main}} E_{Main}$ is associated with the interval $[i \cdot 60\,\text{ms} - \epsilon, (i+1) \cdot 60\,\text{ms} - \epsilon]$. The controller M_{Main} performs a transition using the roll angle observed at the sampling time $c_{Main}(i) + 3\,\text{ms}$. For $u_I = (c_{Main}(i) + 3\,\text{ms}) - (i \cdot 60\,\text{ms} - \epsilon)$, $\boldsymbol{v} = (v_\phi, v_p, v_{\delta_L}, v_{\delta_R})$, $\boldsymbol{v}' = (v'_\phi, v'_p, v'_{\delta_L}, v'_{\delta_R})$, and $\boldsymbol{\tau} = (\tau_\phi, \tau_p, \tau_{\delta_L}, \tau_{\delta_R})$:

$$\begin{aligned} & \big(((g_\phi, v_L, v_R), (i, \boldsymbol{v})), ((i', \boldsymbol{v}'), (v_\phi, g_L, g_R))\big) \in \delta_{M_{Main} \restriction_{\pi_{Main}} E_{Main}} \\ \iff\ & \exists \boldsymbol{\tau} \in \mathcal{R}^{60\,\text{ms}}_{E_{Main}}(*, 10\,\text{ms}, *).\ \ \boldsymbol{v} = \boldsymbol{\tau}(0) \ \ \wedge \ \ \boldsymbol{v}' = \boldsymbol{\tau}(60\,\text{ms}) \ \wedge \\ & \quad \big(((g_\phi, v_L, v_R), (i, \tau_\phi(u_I))), ((i', v'_\phi), (v_\phi, g_L, g_R))\big) \in \delta_{Main}. \end{aligned}$$

The k-step deceleration $(M \restriction E_M)^{\times k}$ of the environment restriction $M \restriction E_M$ can also be defined in a straightforward way. A transition of $(M \restriction E_M)^{\times k}$ consists of k successive transitions of $M \restriction E_M$, and its realizable trajectories are just a concatenation of the k realizable trajectories for the k successive transitions. The definition of "one-step" realizable trajectories can easily be extended for such k-steps of realizable trajectories as follows:

Definition 13. *Trajectories $\boldsymbol{\tau} \in \mathcal{T}^l$ of duration T are k-*step realizable *for control commands $a_0, a_1, \ldots, a_k \in C$ and response durations $u^1_R, \ldots, u^k_R \in \mathbb{R}$, denoted by $\boldsymbol{\tau} \in \mathcal{R}^T_{E_M}(a_0, u^1_R, a_1, \ldots, u^k_R, a_k)$, iff for $u_{R_0} = 0$ and $u_{R_{k+1}} = T$: $\bigwedge_{j=0}^{k} \big((a_j, \boldsymbol{\tau}(u_{R_j}), u_{R_{j+1}} - u_{R_j}), \boldsymbol{\tau} \trianglerighteq u_{R_j} \trianglelefteq u_{R_{j+1}}\big) \in \Lambda$.*

3.4 Hybrid PALS Synchronous Models

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

Definition 14. *A* hybrid multirate ensemble *is a triple $\mathcal{E} \restriction_\Pi E_\mathcal{E}$, composed of: (i) a multirate ensemble $\mathcal{E}$, (ii) a family of interface functions $\Pi = \{\pi_j\}_{j \in J_S \cup J_F}$ (with J_S an index set of slow machines and J_F an index set of fast machines), and (iii) a family of local physical environments $E_\mathcal{E} = \langle \{E_{M_j}\}_{j \in J_S \cup J_F}, (\forall t)\, \psi \rangle$ with $(\forall t)\, \psi$ the time-invariant constraints.*

Example 4. For the simple CPS controller to roll an airplane in Example 1, the multirate ensemble $\mathcal{E} = (J_S, J_F, e, \{M_j\}_{j\in J_s\cup J_F}, E, src, rate, adap)$ is defined by: (i) $J_S = \{Main\}$ and $J_F = \{L, R\}$; (ii) M_{Main} in Example 3, and M_L and M_R in Example 2; (iii) $E = (\mathbb{R}, \mathbb{R})$ for $(v_\phi, goal_\phi)$; (iv) *src* given as Fig. 5; (v) $rate(Main) = 1$ and $rate(L) = rate(R) = 4$; and (vi) $adap(Main)$ taking the last value from a 4-tuple, and $adap(L)$ and $adap(R)$ giving (v, v, v, v) from a single value v. The interface functions $\Pi = \{\pi_{Main}, \pi_L, \pi_R\}$ and the physical environments $\{E_{Main}, E_L, E_R\}$ are defined in Examples 2–3. The time-invariant constraint $(\forall t)\,\psi \equiv (\forall t)\,(\delta_L(t) = x_L(t)) \wedge (\delta_R(t) = x_R(t))$ is given in Example 1. The resulting hybrid ensemble is then $\mathcal{E} \restriction_\Pi \langle\{E_{Main}, E_L, E_R\}, (\forall t)\,\psi\rangle$.

A hybrid multirate ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ induces a normal multirate ensemble $\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ 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 $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ are a *subset* of the behaviors of $\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$, 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 $\boldsymbol{\tau}$ for $[iT - \epsilon, (i+1)T - \epsilon]$ for a global period T. Hence, a lockstep composition of such environment-restricted transitions *whose realizable trajectories* $\boldsymbol{\tau}$ *satisfy the time-invariant constraints* $(\forall t)\,\psi$ gives a transition of the synchronous composition $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ from one state at time $iT - \epsilon$ to another state at time $(i+1)T - \epsilon$.

Definition 15. *Given a hybrid multirate ensemble* $\mathcal{E} \restriction_\Pi E_\mathcal{E}$*, the* synchronous composition *of* $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ *is the typed machine* $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}} = (D_i^\mathcal{E}, S^\mathcal{E}, D_o^\mathcal{E}, \delta_{\mathcal{E}\restriction_\Pi E_\mathcal{E}})$*, where (i)* $D_i^\mathcal{E} = D_o^e$ *and* $D_o^\mathcal{E} = D_i^e$*, (ii)* $S^\mathcal{E} = \prod_{j\in J}((S_j \times \mathbb{R}^{l_j}) \times D_{OF}^j)$*, and (iii)* $\delta_{\mathcal{E}\restriction_\Pi E_\mathcal{E}} \subseteq (D_i^\mathcal{E} \times S^\mathcal{E}) \times (S^\mathcal{E} \times D_o^\mathcal{E})$ *such that, for* $J = J_S \cup J_F$*:*

$$\big((\boldsymbol{i}, \{(s_j, \boldsymbol{v}_j), \boldsymbol{f}_j\}_{j\in J}), (\{(s'_j, \boldsymbol{v}'_j), fout_l(\boldsymbol{o}'_l)\}_{j\in J}, in_e(\{\boldsymbol{o}'_j\}_{j\in J}))\big) \in \delta_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$$

iff, given $\overline{M_j \restriction_\pi E_{M_j}} = ((M_j \restriction_\pi E_{M_j})^{\times rate(j)})_{adap(j)}$ *for* $j \in J$*:*

1. *for each machine* $l \in J$ *with* $k_l = rate(l)$*:*
   (a) $\big((in_l(\boldsymbol{i}, \{\boldsymbol{f}_j\}_{j\in J}), (s_l, \boldsymbol{v}_l)), ((s'_l, \boldsymbol{v}'_l), \boldsymbol{o}'_l)\big) \in \delta_{\overline{M_j \restriction_\pi E_{M_j}}}$
   (b) *for some realizable trajectories* $\boldsymbol{\tau}_l$ *with* $\boldsymbol{\tau}_l(0) = \boldsymbol{v}_l$ *and* $\boldsymbol{\tau}_l(T) = \boldsymbol{v}'_l$*;*[8] *and*
2. *the collection* $\{\boldsymbol{\tau}_l\}_{j\in J}$ *satisfies the time-invariant constraints* $(\forall t)\,\psi$ *in* $E_\mathcal{E}$*;*

where $fout_j(\boldsymbol{o}_j)$ *is the feedback part of output* $\boldsymbol{o}_j$*,* $in_e(\{\boldsymbol{o}'_j\}_{j\in J})$ *is the output to the interface* e *from outputs* $\{\boldsymbol{o}'_j\}_{j\in J}$ *of the machines, and* $in_l(\boldsymbol{i}, \{\boldsymbol{f}_j\}_{j\in J})$ *is the input to* $\overline{M_j \restriction_\pi E_{M_j}}$ *from interface input* $\boldsymbol{i}$ *and feedback outputs* $\{\boldsymbol{f}_j\}_{j\in J}$*.*

Notice that for a multirate ensemble $\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ of $\{M_j \restriction_\pi E_{M_j}\}_{j\in J_S\cup J_F}$, only the condition (1a) is necessary to perform synchronous transitions, and (1b) and (2) further restrict transitions for a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$.

[8] $\boldsymbol{\tau}_l \in \mathcal{R}^T_{E_{M_l}}(\pi_C(s_0), u_{R_1}, \pi_C(s_1), \ldots, u_{R_{k_l}}, \pi_C(s_{k_l}))$ are k_l-step realizable trajectories with $s_0 = s_l$, $s_{k_l} = s'_l$, some intermediate states $s_1, \ldots, s_{k_l-1}$ for k-step deceleration, and $u_{R_i} = (c_{M_l}(\pi_T(s_{i-1})) + \pi_{R_l}(s_{i-1})) - (\pi_T(s_{i-1})(T/k_l) - \epsilon)$ for $1 \le i \le k_l$.

3.5 Hybrid PALS Distributed Models

Hybrid PALS maps a hybrid multirate ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ with a global period T, together with PALS bounds Γ, to the hybrid system $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$. The distributed components in $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are exactly the same as $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ given by the wrapper hierarchies in Fig. 2, but the controllers in $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are also interact with their physical environments in $E_\mathcal{E}$, according to the *sampling and response timing policy* Π that determine sensor sampling timing t_I and actuator response timing t_R for each controller.

The behaviors of the Hybrid PALS distributed system $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are the subset of those of the PALS distributed system $\mathcal{MA}(\mathcal{E}, T, \Gamma)$ that can be realized by the physical environments and the time-invariant constraints in $E_\mathcal{E}$. The continuous dynamics of $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ is "completely" decided by the controllers, their physical environments, the timing policies, and the local clocks of the controllers; that is, it abstracts from asynchronous communication, network delays, message buffering, backoff timers, etc.

More precisely, recall that a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ induces an ensemble $\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ composed of the environment restrictions $\{M_j \restriction_\pi E_{M_j}\}_{j \in J_S \cup J_F}$. The behaviors of $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are the behaviors of $\mathcal{MA}(\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}, T, \Gamma)$ that are restricted by the time-invariant constraints. As explained in Section 2.3, for $\mathcal{MA}(\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}, T, \Gamma)$, *big-step transitions* are defined from one stable state at time $iT - \epsilon$ to another stable state at $(i+1)T - \epsilon$. For such time intervals $[iT - \epsilon, (i+1)T - \epsilon]$, each environment-restricted controller $M_j \restriction_\pi E_{M_j}$ provides (k-step) realizable trajectories $\boldsymbol{\tau}_j$, based on control commands, sampling timings, response timings, and round numbers. Therefore, big-step stable transitions of $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ are exactly those of $\mathcal{MA}(\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}, T, \Gamma)$ whose associated trajectories $\boldsymbol{\tau}_j$ satisfy the time-invariant constraints $(\forall t)\, \psi$.

The correctness of Hybrid PALS immediately follows from the fact that all physical measurements and physical activation *happen at the same time* in both $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$ with the *same timing policies* Π.

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

Proof. Suppose that there exists a transition $s \longrightarrow_{\mathcal{E} \restriction_\Pi E_\mathcal{E}} s'$ by $M_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ and s ($\sim_{obi}$; *sync*) C for a stable state C of $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$. By definition, there exists a transition $s \longrightarrow_{\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}} s'$ by $M_{\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}}$ with some realizable trajectories $\boldsymbol{\tau}$ that satisfy the time-invariant constraints $(\forall t)\, \psi$. By Theorem 1, ($\sim_{obi}$; *sync*) is a bisimulation between $ts(M_{\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}})$ and $ts(Stable(\mathcal{MA}(\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}, T, \Gamma)))$, and thus for some stable state C', there exists a big-step transition $C \longrightarrow_{st} C'$ by $\mathcal{MA}(\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}, T, \Gamma)$ such that s' ($\sim_{obi}$; *sync*) C', where both $s \longrightarrow_{\overline{\mathcal{E} \restriction_\Pi E_\mathcal{E}}} s'$ and $C \longrightarrow_{st} C'$ involve exactly the same machine states and inputs. By construction, control commands, sampling timings, response timings, and round numbers all depend on machine states and inputs. Therefore, $\boldsymbol{\tau}$, which satisfy $(\forall t)\, \psi$, are also realizable for $C \longrightarrow_{st} C'$. Consequently, there exists a big-step transition $C \longrightarrow_{st} C'$ by $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_\Pi E_\mathcal{E}$. The other direction is similar. □

Given a set of atomic propositions AP, consider a state labeling function $\mathcal{L} : S^{\mathcal{E}} \times D_i^{\mathcal{E}} \to 2^{AP}$ for $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$. If $\mathcal{L}$ cannot distinguish $\sim_{obi}$-equivalent states (i.e., $s \sim_{obi} s' \implies \mathcal{L}(s, \boldsymbol{i}) = \mathcal{L}(s', \boldsymbol{i})$), then Theorem 2 implies that $(\sim_{obi}; sync)$ is also a bisimulation between Kripke structures for $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_{\Pi} E_{\mathcal{E}}$. Therefore, they satisfy the same CTL^* formulas.

Corollary 1. *For a hybrid ensemble $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$ and its state labeling function $\mathcal{L} : S^{\mathcal{E}} \times D_i^{\mathcal{E}} \to 2^{AP}$ that cannot distinguish $\sim_{obi}$-equivalent states, the Kripke structures for $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$ and $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_{\Pi} E_{\mathcal{E}}$ satisfy the same set of CTL^* formulas (for $(\sim_{obi}; sync)$-related initial states and the same initial input).*

4 HYBRID PALS MODELS IN SMT

This section first show how analysis problems for a hybrid multirate ensemble $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$, for *all possible local clocks*, can be encoded as logical formulas over the real numbers and ODEs. Theorem 2 implies that a distributed hybrid model $\mathcal{MA}(\mathcal{E}, T, \Gamma) \restriction_{\Pi} E_{\mathcal{E}}$ can then be verified by checking the satisfiability of these formulas for the much simpler synchronous model $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$. It is worth nothing that hybrid automata can also be used (see Appendix A), but it is quite complex since Hybrid PALS models are time-triggered and consider 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 logical formulas over the real numbers is undecidable for nonlinear hybrid systems, but becomes *decidable* up to any given precision $\delta > 0$ [8, 10]. A *δ-complete decision procedure* for a formula ϕ returns false if ϕ is unsatisfiable, and returns true if its *syntactic numerical perturbation* of ϕ by bound δ is satisfiable.[9] This is very useful in practice, since sampling exact physical values is not possible in reality.

4.1 Encoding Hybrid PALS Models

Transition of the synchronous composition $M_{\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}}$ of an ensemble $\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}$ are expressed as formulas of the form $\phi^T_{\mathcal{E} \restriction_{\Pi} E_{\mathcal{E}}}(\boldsymbol{i}, \boldsymbol{y} \shortmid \boldsymbol{y'}, \boldsymbol{o} \shortmid \boldsymbol{\tau})$, where: (i) $\boldsymbol{i}$ denotes the interface inputs, (ii) $\boldsymbol{y}$ denotes the states at the beginning of the round at time $iT - \epsilon$, (iii) $\boldsymbol{y'}$ denotes the states at the end of the round at time $(i+1)T - \epsilon$, (iv) $\boldsymbol{o}$ denotes the interface outputs, and (v) $\boldsymbol{\tau}$ *unary function symbols over time* denoting the trajectories of duration T for the physical parameters in $E_{\mathcal{E}}$.

Encoding Typed Machines. A controller M can be expressed as a logic formula of the form $\phi_M(\boldsymbol{i}, \boldsymbol{y} \shortmid \boldsymbol{y'}, \boldsymbol{o})$, with variables $\boldsymbol{i}$, $\boldsymbol{y}$, $\boldsymbol{y'}$, and $\boldsymbol{o}$ denoting, respectively, the input, the current state, the next state, and the output. That is, $\phi_M(\boldsymbol{i}, s \shortmid s', \boldsymbol{o}) \iff ((\boldsymbol{i}, s), (s', \boldsymbol{o})) \in \delta_M$.

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

Example 5. For an aileron subcontroller M in Example 2, the logical formula $\phi_M(y_{g_M}, y_i, y_{rate_M}, y_{v_M} \shortmid y_i', y_{rate_M}', y_{v_M}', y_o)$ can be defined as the conjunction: $y_{rate_M}' = (y_{g_M} - y_{v_M})/15 \ \wedge \ y_o = y_{v_M} \ \wedge \ y_i' = y_i + 1$.

For the main controller M_{Main} to govern the subcontrollers in Example 3, the logical formula $\phi_{Main}(y_{g_\phi}, y_{v_L}, y_{v_R}, y_i, y_{v_\phi} \shortmid y_i', y_{v_\phi}', y_o, y_{g_L}, y_{g_R})$ is defined by: $y_{g_R} = 0.3 \cdot (y_{g_\phi} - y_{v_\phi}) \ \wedge \ y_{g_L} = -y_{g_R} \ \wedge \ y_o = y_{v_\phi} \ \wedge \ y_i' = y_i + 1$.

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

If the continuous dynamics of $\boldsymbol{x}$ is specified as ODEs $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = F_{\boldsymbol{a}}(\boldsymbol{x}, t)$ for a control command $\boldsymbol{a}$ and a time interval $[u_0, u_t]$, then ϕ_{E_M} includes universal quantification over time along with solutions of the ODEs, such as a formula of the form: $\bigvee \big(guard(\boldsymbol{a}) \rightarrow \forall t \in [u_0, u_t].\ \boldsymbol{\tau}(t) = \boldsymbol{v} + \int_0^{t-u_0} F_{\boldsymbol{a}}(\boldsymbol{x}, t)\,\mathrm{d}t\big)$.

Example 6. For the physical environment E_M of a subcontroller M in Example 1, the formula $\phi_{E_M}(y_{rate_M}, y_{v_M}, y_{u_0}, y_{u_t} \shortmid \tau_M)$ is defined by:

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

For the physical environment E_{Main} of the main controller M_{Main}, the formula $\phi_{E_{Main}}(y_{v_\phi}, y_{v_p}, y_{v_{\delta_L}}, y_{v_{\delta_R}}, y_{u_0}, y_{u_t} \shortmid \tau_\phi, \tau_p, \tau_{\delta_L}, \tau_{\delta_R})$ is defined by:

$$\forall t \in [y_{u_0}, y_{u_t}].\ \begin{bmatrix} \tau_\phi \\ \tau_p \end{bmatrix}(t) = \begin{bmatrix} y_{v_\phi} \\ y_{v_p} \end{bmatrix} + \int_0^{t-y_{u_0}} \begin{bmatrix} \tau_p(t) \\ c(\tau_{\delta_R}(t) - \tau_{\delta_L}(t)) \end{bmatrix} \mathrm{d}t.$$

Encoding Environment Restrictions. A environment restriction $M \restriction_\pi E_M$ is encoded as a formula of the form $\phi^{T,i}_{M\restriction_\pi E_M}(\boldsymbol{i}, \boldsymbol{y}, \boldsymbol{v} \shortmid \boldsymbol{y'}, \boldsymbol{v'}, \boldsymbol{o} \shortmid \boldsymbol{\tau})$, with unary function symbols $\boldsymbol{\tau}$ denoting E_M's trajectories, and variables: (i) $\boldsymbol{i}$ denoting input, (ii) $(\boldsymbol{y}, \boldsymbol{v})$ denoting a state at the beginning of the round, (iii) $(\boldsymbol{y'}, \boldsymbol{v'})$ denoting a state at the end of the round, and (iv) $\boldsymbol{o}$ denoting output, given a period $T \in \mathbb{R}$ and a round number $i \in \mathbb{N}$.

The exact 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, since the concrete values of the imprecise local clock $c_M(i)$ are determined on-the-fly by *clock synchronization* mechanisms. 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 $\phi_{M\restriction_\pi E_M}(\boldsymbol{i}, s, \boldsymbol{v} \shortmid s', \boldsymbol{v'}, \boldsymbol{o})$ iff $((\boldsymbol{i}, (s, \boldsymbol{v})), ((s', \boldsymbol{v'}), \boldsymbol{o})) \in \delta_{M\restriction_\pi E_M}$ for some c_M.

Definition 16. *For an environment restriction $M \restriction_\pi E_M$ with interface π, the formula $\phi^{T,i}_{M\restriction_\pi E_M}(\boldsymbol{i}, \boldsymbol{y}, \boldsymbol{v} \shortmid \boldsymbol{y'}, \boldsymbol{v'}, \boldsymbol{o} \shortmid \boldsymbol{\tau})$ is defined by:*[10]

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

Example 7. For an environment-restricted subcontroller $M \restriction_\pi E_M$ in Example 2, the formula $\phi^{15,0}_{M\restriction_\pi E_M}(y_{g_M}, y_i, y_{rate_M}, y_{v_M} \shortmid y'_i, y'_{rate_M}, y'_{v_M}, y_o \shortmid \tau_M)$ is defined by:

$$\begin{aligned}(\exists u_I, u_R, v_I, v_R)\ \ 1 < u_I < 1 + 2\epsilon \ \ &\wedge \ \ 2 < u_R < 2 + 2\epsilon \ \ \wedge \\ v_I = \tau_M(u_I) \ \ \wedge \ \ v_R = \tau_M(u_R) \ \ &\wedge \ \ y_{v_M} = \tau_M(0) \ \ \wedge \ \ y'_{v_M} = \tau_M(15) \ \ \wedge \\ \phi_{E_M}(y_{rate_M}, y_{v_M}, 0, u_R \shortmid \tau_M) \ \ &\wedge \ \ \phi_{E_M}(y'_{rate_M}, v_R, u_R, 15, \shortmid \tau_M) \ \ \wedge \\ &\phi_M(y_{g_M}, y_i, y_{rate_M}, v_I \shortmid y'_i, y'_{rate_M}, y'_{v_M}, y_o).\end{aligned}$$

For the environment restriction $M_{Main} \restriction_{\pi_{Main}} E_{Main}$ in Example 3, the formula $\phi^{60,0}_{M_{Main}\restriction_{\pi_{Main}} E_{Main}}(y_{g_\phi}, y_{v_L}, y_{v_R}, y_i, \boldsymbol{y} \shortmid y'_i, \boldsymbol{y'}, y_o, y_{g_L}, y_{g_R} \shortmid \boldsymbol{\tau})$ is defined by:

$$\begin{aligned}(\exists u_I, v^\phi_I)\ \ 3 < u_I < 3 + 2\epsilon \ \ &\wedge \ \ v^\phi_I = \tau_\phi(u_I) \ \ \wedge \ \ \boldsymbol{y} = \boldsymbol{\tau}(0) \ \ \wedge \ \ \boldsymbol{y'} = \boldsymbol{\tau}(60) \ \ \wedge \\ \phi_{E_{Main}}(\boldsymbol{y}, 0, 60 \shortmid \boldsymbol{\tau}) \ \ &\wedge \ \ \phi_{Main}(y_{g_\phi}, y_{v_L}, y_{v_R}, y_i, v^\phi_I \shortmid y'_i, y'_{v_\phi}, y_o, y_{g_L}, y_{g_R}).\end{aligned}$$

for $\boldsymbol{y} = (y_{v_\phi}, y_{v_p}, y_{v_{\delta_L}}, y_{v_{\delta_R}})$, $\boldsymbol{y'} = (y'_{v_\phi}, y'_{v_p}, y'_{v_{\delta_L}}, y'_{v_{\delta_R}})$, and $\boldsymbol{\tau} = (\tau_\phi, \tau_p, \tau_{\delta_L}, \tau_{\delta_R})$ Notice that there are no formulas for response behaviors in $\phi^{60,0}_{M_{Main}\restriction_{\pi_{Main}} E_{Main}}$, since E_{Main} does not have control commands.

Encoding k-Step Decelerations. 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 defined by sequentially composing the k formulas $\phi^{T/k,ik}_{M\restriction_\pi E_M}, \ldots, \phi^{T/k,(ik+k-1)}_{M\restriction_\pi E_M}$ corresponding to the k *subintervals* $[(ik+n-1)T/k - \epsilon, (ik+n)T/k - \epsilon]$ for $n = 1, \ldots, k$.

Definition 17. *Given a environment restriction $M \restriction_\pi E_M$ and $k \in \mathbb{N}$, the formula $\phi^{T,i}_{(M\restriction_\pi E_M)^{\times k}}(\langle \boldsymbol{i_1}, \ldots, \boldsymbol{i_k}\rangle, \boldsymbol{y_0}, \boldsymbol{v_0} \shortmid \boldsymbol{y}_k, \boldsymbol{v}_k, \langle \boldsymbol{o_1}, \ldots, \boldsymbol{o_k}\rangle \shortmid \boldsymbol{\tau})$ is:*

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

Example 8. The 4-step deceleration of $M \restriction_\pi E_M$ in Example 2 is encoded as the logical formula $\phi^{60,i}_{(M\restriction_\pi E_M)^{\times 4}}(\langle y^1_{g_M}, \ldots, y^4_{g_M}\rangle, \boldsymbol{y}_0, y^0_{v_M} \shortmid \boldsymbol{y}_0, y^4_{v_M}, \langle y^1_o, \ldots, y^4_o\rangle \shortmid \tau_M)$ using the formula $\phi^{15,i}_{M\restriction_\pi E_M}$ in Example 7, where $\boldsymbol{y}_n = (y^n_i, y^n_{rate_M})$, such that:

$$\exists \{\boldsymbol{y_n}, y^n_{v_M}\}_{n=1}^{3}.\ \bigwedge_{n=1}^{4} \phi^{15,(4i+n-1)}_{M\restriction_\pi E_M}(y^n_{g_M}, \boldsymbol{y}_{n-1}, y^{n-1}_{v_M} \shortmid \boldsymbol{y_n}, y^n_{v_M}, y^n_o \shortmid \tau_M)$$

[10] Instead of $iT - \epsilon$ and $(i+1)T - \epsilon$, we use the "ϵ-shifted" time axis for $[iT, (i+1)T]$.

Encoding Wiring Diagrams. The wiring diagram of an ensemble $\mathcal{E}$ is encoded as a conjunction of appropriate equalities between variables denoting input and output ports. Each equality corresponds to a connection in $\mathcal{E}$ (together with an input adaptor for typed 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 $\mathcal{E}$.

Definition 18. *The formula ϕ_{wire} is the conjunction of the equalities, composed of: (i) $\alpha^l_j(i^l_j) = f^n_k$ and $f^{n\prime}_k = o^n_k$ for connection from M_k's n-th output port to M_j's l-th input port with its adaptor α^l_j, where f^l_k denotes a feedback output from the previous step, and $f^{l\prime}_k$ denotes one for the next step; (ii) $\alpha^l_j(i^l_j) = i^n_e$ for connection from $\mathcal{E}$'s n-th input port to M_j's l-th input port with adaptor α^l_j; (iii) $o^n_e = o^l_j$ for connection from M_j's l-th output port to $\mathcal{E}$'s n-th output port.*

Example 9. For the ensemble $\mathcal{E}$ in Example 4 for the simple CPS controller in Example 1, the formula ϕ_{wire} is defined as follows (where different variables with the same names are appropriately distinguished using superscripts):

$$\begin{aligned}(y^{Main}_{v_L} = f^4_{o_L} \ \wedge\ f^{4\prime}_{o_L} = y^4_{o_L}) \ \wedge\ (\textstyle\bigwedge_{i=1}^4 y^i_{g_L} = f^{Main}_{g_L} \ \wedge\ f^{Main\prime}_{g_L} = y^{Main}_{g_L})\\ \wedge\ (y^{Main}_{v_R} = f^4_{o_R} \ \wedge\ f^{4\prime}_{o_R} = y^4_{o_R}) \ \wedge\ (\textstyle\bigwedge_{i=1}^4 y^i_{g_R} = f^{Main}_{g_R} \ \wedge\ f^{Main\prime}_{g_R} = y^{Main}_{g_R})\\ \wedge\ (y^{Main}_{g_\phi} = y^{\mathcal{E}}_{g_\phi} \ \wedge\ y^{\mathcal{E}}_{v_\phi} = y^{Main}_{v_\phi}).\end{aligned}$$

Encoding Hybrid Ensembles. A hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ is encoded as a formula $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\boldsymbol{i}, \{\boldsymbol{z_j}, \boldsymbol{f_j}\}_{j\in J_S\cup J_F} \shortmid \{\boldsymbol{z'_j}, \boldsymbol{f'_j}\}_{j\in J_S\cup J_F}, \boldsymbol{o} \shortmid \{\boldsymbol{\tau}_j\}_{j\in J_S\cup J_F})$, with unary function symbols $\boldsymbol{\tau}_j$ denoting trajectories for physical environments, and variables $\boldsymbol{i}$, $\boldsymbol{z_j}$, $\boldsymbol{f_j}$, $\boldsymbol{z'_j}$, $\boldsymbol{f'_j}$, and $\boldsymbol{o}$ denoting, respectively, inputs, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the beginning of the round, feedback outputs from the previous round, the state of $M_j \restriction_{\pi_j} E_{M_j}$ at the end of the round, the feedback outputs for the next round, and the output.

Definition 19. *For a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ of machines $\{M_j\}_{j\in J_S\cup J_F}$, their physical environments $\{E_{M_j}\}_{j\in J_S\cup J_F}$ and the time-invariant constraint $(\forall t.\ \psi)$, $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\boldsymbol{i}, \{\boldsymbol{y_j}, \boldsymbol{v_j}, \boldsymbol{f_j}\}_{j\in J_S\cup J_F} \shortmid \{\boldsymbol{y'_j}, \boldsymbol{v'_j}, \boldsymbol{f'_j}\}_{j\in J_S\cup J_F}, \boldsymbol{o} \shortmid \{\boldsymbol{\tau}_j\}_{j\in J_S\cup J_F})$ is:*

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

Example 10. Consider the hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ in Example 4. For variables $\boldsymbol{y} = (y_{v_\phi}, y_{v_p}, y_{v_{\delta_L}}, y_{v_{\delta_R}})$, $\boldsymbol{y'} = (y'_{v_\phi}, y'_{v_p}, y'_{v_{\delta_L}}, y'_{v_{\delta_R}})$, $\boldsymbol{\tau} = (\tau_\phi, \tau_p, \tau_{\delta_L}, \tau_{\delta_R})$, and

$\boldsymbol{y}_n^m = (y_{i_m}^n, y_{rate_m}^n)$ for $m \in \{L, R\}$ and $1 \le n \le 4$:

$$\begin{array}{l}
\phi^{60}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(y^{\mathcal{E}}_{g_\phi}, (y_i, \boldsymbol{y}, f^{Main}_{g_L}, f^{Main}_{g_R}), (\boldsymbol{y}^L_0, y^0_{v_L}, f^4_{o_L}), (\boldsymbol{y}^R_0, y^0_{v_R}, f^4_{o_R}) \mid \\
\quad (y_i', \boldsymbol{y}', f^{Main'}_{g_L}, f^{Main'}_{g_R}), (\boldsymbol{y}^L_4, y^4_{v_L}, f^{4\prime}_{o_L}), (\boldsymbol{y}^R_4, y^4_{v_R}, f^{4\prime}_{o_R}), y^{\mathcal{E}}_{v_\phi} \mid \boldsymbol{\tau}, \tau_L, \tau_R) \\
\Updownarrow \\
\phi^{60,0}_{M_{Main}\restriction_{\pi_{Main}} E_{Main}}(y^{Main}_{g_\phi}, y^{Main}_{v_L}, y^{Main}_{v_R}, y_i, \boldsymbol{y} \mid y_i', \boldsymbol{y}', y^{Main}_o, y^{Main}_{g_L}, y^{Main}_{g_R} \mid \boldsymbol{\tau}) \\
\qquad \wedge\ \phi^{60,0}_{(M_L\restriction_\pi E_L)^{\times 4}}(\langle y^1_{g_L}, \ldots, y^4_{g_L}\rangle, \boldsymbol{y}^L_0, y^0_{v_L} \mid \boldsymbol{y}^L_4, y^4_{v_L}, \langle y^1_{o_L}, \ldots, y^4_{o_L}\rangle \mid \tau_L) \\
\qquad \wedge\ \phi^{60,0}_{(M_R\restriction_\pi E_R)^{\times 4}}(\langle y^1_{g_R}, \ldots, y^4_{g_R}\rangle, \boldsymbol{y}^R_0, y^0_{v_R} \mid \boldsymbol{y}^R_4, y^4_{v_R}, \langle y^1_{o_R}, \ldots, y^4_{o_R}\rangle \mid \tau_R) \\
\qquad \wedge\ \phi_{wire}\ \wedge\ (\forall t.\ \tau_{\delta_L}(t) = \tau_L(t)\ \wedge\ \tau_{\delta_R}(t) = \tau_R(t)).
\end{array}$$

By construction, a formula $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable iff there is a corresponding transition of the synchronous composition $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ for *some* local clocks. Hence, by the bisimulation equivalence (Theorem 2):

Theorem 3. *$\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ is satisfiable iff there is a corresponding* stable transition *for some local clocks in $\mathcal{MA}(\mathcal{E}, T, \Gamma)\restriction_\Pi E_\mathcal{E}$.*

4.2 Encoding Verification Problems

Our goal is to verify safety properties of a distributed CPS $\mathcal{MA}(\mathcal{E}, T, \Gamma)\restriction_\Pi E_\mathcal{E}$. We exploit the bisimulation equivalence $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}} \approx \mathcal{MA}(\mathcal{E}, T, \Gamma)\restriction_\Pi E_\mathcal{E}$ (by Theorem 2) to verify the distributed hybrid system $\mathcal{MA}(\mathcal{E}, T, \Gamma)\restriction_\Pi E_\mathcal{E}$ by using the simpler hybrid system $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$. A safety property is expressed as a formula of the form $safe(\boldsymbol{y}, \boldsymbol{\tau}(t))$ for *state variables* $\boldsymbol{y}$, *trajectories* $\boldsymbol{\tau}$, and time variable t. A transition of $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ for a global period $[iT - \epsilon, (i+1)T - \epsilon]$ is encoded as a formula $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\boldsymbol{i}, \boldsymbol{y} \mid \boldsymbol{y'}, \boldsymbol{o} \mid \boldsymbol{\tau})$. Using these formulas *safe* and $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ as building blocks, This section shows how standard verification problems for hybrid systems, such as bounded reachability, inductive reasoning, and compositional assume-guarantee reasoning, can be encoded as formulas.

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 $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ as an SMT formula. If the formula is unsatisfiable (i.e., no counterexample exists), then, by Theorem 3, the system satisfies the safety property in $[-\epsilon, nT - \epsilon]$ *for any local clocks.*

Definition 20. *A bounded reachability problem of $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ for a safety property $safe(\boldsymbol{y}, \boldsymbol{\tau}(t))$* up to $n \in \mathbb{N}$ rounds *with an initial condition $init(\boldsymbol{y})$ and an input constraint $in(\boldsymbol{i})$ is encoded by:*

$$\exists \boldsymbol{y}_0, \{\boldsymbol{y}_k, \boldsymbol{i}_k, \boldsymbol{o}_k, t_k\}_{k=1}^n.\ init(\boldsymbol{y}_0)\ \wedge\ \bigwedge_{k=1}^{n} \left(\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}(\boldsymbol{i}_k, \boldsymbol{y}_{k-1} \mid \boldsymbol{y}_k, \boldsymbol{o}_k \mid \boldsymbol{\tau}_k) \wedge in(\boldsymbol{i}_k)\right) \wedge \bigvee_{k=1}^{n} \neg safe(\boldsymbol{y}_k, \boldsymbol{\tau}_k(t_k)).$$

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

Example 11. For the simple CPS controller in Example 1, consider the safety property that the roll angle ϕ is always in a given bound $[-\gamma, \gamma]$. As a formula, $\mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t)) \equiv (\mathrm{abs}(v_\phi) \leq \gamma \wedge \mathrm{abs}(\tau_\phi(t)) \leq \gamma)$. Let $\mathit{init}(\boldsymbol{y}) \equiv (\boldsymbol{y} = \boldsymbol{0})$, and $\mathit{in}(y^\mathcal{E}_{g_\phi}) \equiv (y^\mathcal{E}_{g_\phi} = 0)$. Using the formula $\phi^{60}_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}$ in Example 10, the n-step bounded reachability problem can be encoded according to Definition 20.

Inductive Reasoning. For *unbounded* time verification of a safety property $\mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t))$, we encode its inductive proof as a logical formula by using an inductive condition for $M_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}$'s synchronous transitions, which implies the safety property. Such an inductive invariant consists of two formulas: (i) $\mathit{ind}_d(\boldsymbol{y})$ for state variables $\boldsymbol{y}$, and (ii) $\forall t \in [0, T].\, \mathit{ind}_c(\boldsymbol{\tau}(t))$ for trajectories $\boldsymbol{\tau}$.[11]

Definition 21. *An inductive analysis problem of $M_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}$ for a safety property $\mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t))$, an initial condition $\mathit{init}(\boldsymbol{y})$, and an input constraint $\mathit{in}(\boldsymbol{i})$, using an inductive invariant $(\forall t.\ \mathit{ind}(\boldsymbol{y}, \boldsymbol{\tau}(t))) \equiv (\mathit{ind}_d(\boldsymbol{y}) \ \wedge\ \forall t \in [0, T].\, \mathit{ind}_c(\boldsymbol{\tau}(t)))$, where time variable t does not occur in $\mathit{ind}_d(\boldsymbol{y})$, is encoded by:*

- $\forall \boldsymbol{y}.\ \mathit{init}(\boldsymbol{y}) \implies \mathit{ind}_d(\boldsymbol{y})$
- $\forall \boldsymbol{y}, \boldsymbol{y'}, \boldsymbol{i}, \boldsymbol{o}.\ (\mathit{ind}_d(\boldsymbol{y}) \wedge \mathit{in}(\boldsymbol{i}) \wedge \phi^T_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}(\boldsymbol{i}, \boldsymbol{y} \frown \boldsymbol{y'}, \boldsymbol{o} \frown \boldsymbol{\tau})) \implies \forall t.\ \mathit{ind}(\boldsymbol{y}, \boldsymbol{\tau}(t))$
- $\forall \boldsymbol{y}.\ (\forall t \in [0, T].\, \mathit{ind}(\boldsymbol{y}, \boldsymbol{\tau}(t))) \implies \forall t \in [0, T].\, \mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t))$

These formula encode the classical inductive analysis approach. The *init* condition implies the ind_d condition for any state variables $\boldsymbol{y}$. If a transition of $M_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}$ is taken from state $\boldsymbol{y}$ satisfying the ind_d condition, then: (i) the ind_d condition again holds for any next state $\boldsymbol{y'}$, and (ii) in the meantime the ind_c condition holds for trajectories $\boldsymbol{\tau}$. The inductive invariant $\forall t.\, \mathit{ind}(\boldsymbol{y}, \boldsymbol{\tau}(t))$ implies the safety property $\forall t \in [0, T].\, \mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t))$ for one round. By proving the these conditions, we show that the safety property holds for unbounded time with unbounded number of transitions.The formulas can be proved by checking the unsatisfiability of their *negated versions* using SMT solvers.[12]

Example 12. For the simple CPS controller in Example 1, consider the safety property $\mathit{safe}(\boldsymbol{y}, \boldsymbol{\tau}(t)) \equiv (\mathrm{abs}(v_\phi) \leq \gamma \wedge \mathrm{abs}(\tau_\phi(t)) \leq \gamma)$, the initial condition $\mathit{init}(\boldsymbol{y}) \equiv (v_\phi = 0) \wedge (v_p = 0) \wedge (v_L = 0) \wedge (\mathit{rate}_L = 0) \wedge (v_R = 0) \wedge (\mathit{rate}_R = 0)$, and the input constraint $\mathit{in}(y^\mathcal{E}_{g_\phi}) \equiv (y^\mathcal{E}_{g_\phi} = 0)$ in Example 11. Using simple inductive formulas $\mathit{ind}_d(\boldsymbol{y}) \equiv (\boldsymbol{y} = \boldsymbol{0})$ and $\mathit{ind}_c(\boldsymbol{\tau}(t)) \equiv (\boldsymbol{\tau}(t) = 0)$, the inductive analysis problem can be encoded according to Definition 21.

[11] One of an important problem is to find such an inductive invariant for $M_{\mathcal{E}\uparrow_\Pi E_\mathcal{E}}$, but providing general solutions for this problem is beyond the scope of this paper.

[12] This may involve universally quantified time variable t for the last condition. For example, the dReal SMT solver can deal with such constraints.

Compositional Reasoning. We encode a divide-and-conquer proof to verify a safety property based on standard assume-guarantee reasoning. In $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$, conceptually, each subcomponent $M_j \restriction_{\pi_j} E_{M_j}$ performs a transition based on its input $\boldsymbol{i}_j$ and trajectories $\boldsymbol{\tau}_j$ that are restricted by time-invariant constraints. Therefore, we consider an input condition $c^j_{in}(\boldsymbol{i}_j, \boldsymbol{\tau}_j(t))$ and an output condition $c^j_{out}(\boldsymbol{o}_j, \boldsymbol{\tau}_j(t))$ for each subcomponent $M_j \restriction_{\pi_j} E_{M_j}$ during a global round such that a collection of output conditions $\{c^j_{out}(\boldsymbol{o}, \boldsymbol{\tau}_j(t))\}_{j \in J_S \cup J_F}$ implies each input condition $c^j_{in}(\boldsymbol{i}_j, \boldsymbol{\tau}_j(t))$, which again implies the safety property $safe_j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t))$ of the subcomponent $M_j \restriction_{\pi_j} E_{M_j}$ for that round.

Definition 22. *For each subcomponent $j \in J_S \cup J_F$ in $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$, consider a safety property $safe_j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t))$, an input constraint $in_j(\boldsymbol{i}_j)$ for input from $\mathcal{E}$'s interface, an initial condition $init_j(\boldsymbol{y}_j)$, and I/O conditions*

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

Let $\overline{M}_j$ denote its decelerated version $(M_j \restriction_{\pi_j} E_{M_j})^{\times rate(j)}$ for a fast component $j \in J_F$, and $M_j \restriction_{\pi_j} E_{M_j}$ for a slow component $j \in J_S$. Necessary constraints on I/O conditions for a compositional analysis of $M_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ are encoded by:

- *I/O correspondence: for each component $j \in J_S$, $\forall \boldsymbol{i}_j, \boldsymbol{o}_j, \boldsymbol{y}_j, \boldsymbol{y}'_j$:*

$$\big(\forall t.\ c^j_{in}(\boldsymbol{i}_j, \boldsymbol{\tau}_j(t)) \wedge \phi^{T,0}_{\overline{M}_j}(\boldsymbol{i}_j, \boldsymbol{y}_j \shortmid \boldsymbol{y}'_j, \boldsymbol{o}_j \shortmid \boldsymbol{\tau}_j)\big) \implies \forall t.\ c^j_{out}(\boldsymbol{o}_j, \boldsymbol{\tau}_j(t))$$

- *Intercomponent correspondence:*

$$\Big(\bigwedge_{j \in J_S \cup J_F} \forall t.\ c^j_{out}(\boldsymbol{o}, \boldsymbol{\tau}_j(t))\Big) \implies \bigwedge_{j \in J_S \cup J_F} \forall t.\ c^j_{in}(\boldsymbol{i}_j, \boldsymbol{\tau}_j(t))$$

We can consider both bounded reachability and inductive analysis problems based on compositional reasoning. A compositional bounded reachability problem up to $n \in \mathbb{N}$ rounds for component j is encoded by:

$$\begin{aligned}&\exists \boldsymbol{y}^j_0, \{\boldsymbol{y}^j_k, \boldsymbol{i}^j_k, \boldsymbol{o}^j_k, t_k\}_{k=1}^n.\ init_j(\boldsymbol{y}^j_0) \wedge \bigwedge_{k=1}^n \phi^{T,0}_{\overline{M}_j}(\boldsymbol{i}^j_k, \boldsymbol{y}^j_{k-1} \shortmid \boldsymbol{y}^j_k, \boldsymbol{o}^j_k \shortmid \boldsymbol{\tau}^j_k)\\ &\wedge \bigwedge_{k=1}^n \big(in_j(\boldsymbol{i}^j_k) \wedge \forall t.\ c^j_{in}(\boldsymbol{i}^j_k, \boldsymbol{\tau}^j_k(t))\big) \wedge \bigvee_{k=1}^n \neg safe_j(\boldsymbol{y}^j_k, \boldsymbol{\tau}^j_k(t_k)).\end{aligned}$$

A compositional inductive analysis problem for component j with an inductive invariant $\forall t.\ ind^j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t)) \equiv ind^j_d(\boldsymbol{y}_j) \wedge \forall t \in [0,T].\ ind^j_c(\boldsymbol{\tau}_j(t))$ is encoded by:

- $\forall \boldsymbol{y}_j.\ init_j(\boldsymbol{y}_j) \implies ind^j_d(\boldsymbol{y}_j)$
- $\forall \boldsymbol{y}_j, \boldsymbol{y}'_j, \boldsymbol{i}_j, \boldsymbol{o}_j.\ (ind^j_d(\boldsymbol{y}_j) \wedge in_j(\boldsymbol{i}_j) \wedge \forall t.\ c^j_{in}(\boldsymbol{i}_j, \boldsymbol{\tau}_j(t)) \wedge \phi^{T,0}_{\overline{M}_j}(\boldsymbol{i}, \boldsymbol{y} \shortmid \boldsymbol{y}', \boldsymbol{o} \shortmid \boldsymbol{\tau}))$
  $\implies \forall t.\ ind_j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t))$

– $\forall \boldsymbol{y}_j.\ (\forall t \in [0,T].\ ind_j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t))) \implies \forall t \in [0,T].\ safe_j(\boldsymbol{y}_j, \boldsymbol{\tau}_j(t))$

Two necessary constraints in this definition state that given I/O conditions are correct assumptions for compositional reasoning: (i) for each component, assuming its input condition, if a transition is taken, then its output condition holds, and (ii) output conditions for subcomponents implies input conditions for their connected components, either by a wiring diagram or by time-invariant constraints. Bounded reachability and inductive analysis can then be separately performed for each component in a compositional way, by additionally assuming input conditions for individual components. The validity of such formulas for compositional analysis can also be automatically checked using SMT solving by checking the unsatisfiability of their *negations* (see Section 5 for examples).

4.3 Removing Universal Quantification

SMT-based techniques for hybrid systems normally use the SMT theory of the real numbers and computable (nonlinear) real functions, such as polynomials, trigonometric functions, and solutions of Lipschitz-continuous ODEs. Notice that solutions of ODEs are considered *atomic functions*. However, formulas encoding Hybrid PALS models may contain ODEs and universal quantification over *uninterpreted functions on the real numbers*, such as a formula $\forall t \in [u_0, u_t].\ \boldsymbol{x}(t) = \boldsymbol{v} + \int_0^{t-u_0} F_{\boldsymbol{a}}(\boldsymbol{x}, t)\,\mathrm{d}t$, or time-invariant constraints ($\forall t.\ \psi$), which are not supported by current state-of-the-art SMT techniques. This section explains how such universal quantification can be removed from the formulas.

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

The basic idea is to assign a *complete system* of ODEs, which combine all partial ODE systems for physical environments in $\mathcal{E} \restriction_\Pi E_\mathcal{E}$, to every time point in a global round. Suppose that a partial ODE system $\frac{\mathrm{d}\boldsymbol{x}_j}{\mathrm{d}t} = F^j_{\boldsymbol{a}_j}(\boldsymbol{x}_j, t)$ is assigned to an interval $[u_j, u'_j]$ in a global round by each environment-restricted controller $M_j \restriction_{\pi_j} E_{M_j}$. A complete ODE system at time z is then given by the ODEs $\{\frac{\mathrm{d}\boldsymbol{x}_j}{\mathrm{d}t} = F^j_{\boldsymbol{a}_j}(\boldsymbol{x}_j, t)\}_{j \in J_S \cup j_F}$, where time z is included in every interval $[u_j, u'_j]$ for $M_j \restriction_{\pi_j} E_{M_j}$ (that is, $z \in [u_j, u'_j]$), provided that variable are accordingly renamed by the equality time-invariant constraints.

Recall that each occurrence of (partial) ODEs in formulas for Hybrid PALS models has the form $\forall t \in [u_0, u_t].\ \boldsymbol{x}(t) = \boldsymbol{v} + \int_0^{t-u_0} F_{\boldsymbol{a}}(\boldsymbol{x}, t)\,\mathrm{d}t$, which means exactly that the (partial) ODE system $F_{\boldsymbol{a}}(\boldsymbol{x}, t)$ is assigned to the interval $[u_0, u_t]$. Since a formula $\phi^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ for one global round of a hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ includes a finite number of such subformulas, there exists only a finite number of combinations of complete ODE systems in one global round. Therefore, it is possible to construct an equivalent formula that only includes a complete

system of ODEs (e.g., transforming $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ to its disjunctive normal form, and then performing case analysis on any possible arrangements of time intervals).

Example 13. For the hybrid ensemble $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ in Example 4, the formula $\phi^{60}_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ in Example 10 involves the following time segments: (i) for the main controller, the ODEs $\dot{\tau_\phi} = \tau_p$ and $\dot{\tau_\phi} = c(\tau_{\delta_R} - \tau_{\delta_L})$ to the interval $[0, 60]$, and (ii) for each subcontroller M, the ODE $\dot{\tau_M} = y^I_{rate^i_M}$ to $[15i, 15i + u^M_{R,i}]$, and $\dot{\tau_M} = y^R_{rate^i_M}$ to $[15i + u^M_{R,i}, 15(i+1)]$ for $i \in \{0, 1, 2, 3\}$. Given an ordering $u^L_{R,i} \leq u^R_{R,i}$ for each i (i.e., the left aileron controller always responds earlier than the right one), then the complete ODE system assigned to the interval $[15i + u^L_{R,i}, 15i + u^R_{R,i}]$ is

$$\dot{\tau_\phi} = \tau_p, \quad \dot{\tau_\phi} = c(\tau_R - \tau_L), \quad \dot{\tau_L} = y^R_{rate^i_L}, \quad \dot{\tau_R} = y^I_{rate^i_R},$$

where τ_{δ_R} and τ_{δ_L} are renamed by $(\forall t.\ \tau_{\delta_L}(t) = \tau_L(t) \ \wedge \ \tau_{\delta_R}(t) = \tau_R(t))$. This ODE system indicates a period that the left subcontroller has responded (with the new rate $y^R_{rate^i_L}$) but the right subcontroller has not responded yet.

We now show how Hybrid PALS models can be encoded using the modular encoding framework illustrated in [5], available for the dReal SMT solver. This modular framework uses *function names* and *parameterized integration operators*. For example, we can have *named* real functions with names $\mathtt{m}_1$ and $\mathtt{m}_2$:

$$(\mathtt{m}_1)\ \ \sin(x(t)) : \mathit{Real} \to \mathit{Real}, \qquad (\mathtt{m}_2)\ \ [y(t) + 1, z(t)^2] : \mathit{Real} \to \mathit{Real}^2,$$

and the the following integration terms constructed by the integral operator *int* and function names, so that solutions of ODEs are *no longer* atomic functions:

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

Basically, we build *symbolic time segments* using name variables, assignments, and parameterized integration terms.

First, a global period $[0, T]$ for one synchronous round of $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ is dividid into N contiguous subintervals $[0, t_1], [t_1, t_2], [t_2, t_3], \ldots, [t_{N-2}, t_{N-1}], [t_{N-1}, T]$ such that $0 \leq t_1 \leq t_2 \leq \cdots \leq t_{N-1} \leq T$. 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 of the form $\forall t \in [u_0, u_t].\ \boldsymbol{x}(t) = \boldsymbol{v} + \int_0^{t-u_0} F_{\boldsymbol{a}}(\boldsymbol{x}, t)\,\mathrm{d}t$ in the formula $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$. Because each ODE subformula involves two time points u_0 and u_t, if there are k such ODE subformulas in $\phi^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$, then the total number of possible time segments with different ODE assignments is $N = 2 \cdot k$.

Second, for each physical environment E_{M_j}, N *name variables* $f^1_j, f^2_j, \ldots, f^N_j$ are declared, which denote the behavior of E_{M_j}'s physical parameters $\boldsymbol{x}_j$ for the N subintervals . Each variable f^i_j denotes a (partial) system of ODEs for corresponding to the interval $[t_{i-1}, t_i]$ (with $t_0 = 0$ and $t_N = T$). As illustrated in Figure 7, an ODE system $F_{\boldsymbol{a}}(\boldsymbol{x}_j, t)$ with name $\mathtt{m}_{F_a(\boldsymbol{x_j}, t)}$ is assigned to *some*

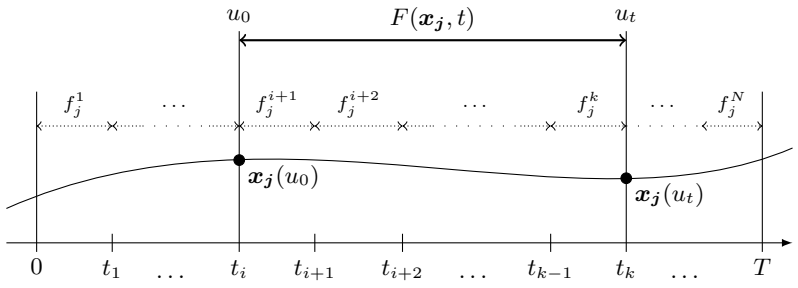


Fig. 7. Time Segments for ODEs

interval $[t_i, t_k]$ by declaring $f_j^l = \mathtt{m}_{F_a(\boldsymbol{x_j},t)}$ for $i < l \leq k$. That is, for an ODE formula $\forall t \in [u_0, u_t].\, \boldsymbol{x}_j(t) = \boldsymbol{v}_j + \int_0^{t-u_0} F_{\boldsymbol{a}}(\boldsymbol{x}_j, t)\,\mathrm{d}t$, we declare:

$$(t_i = u_0) \ \wedge\ (t_k = u_t) \ \wedge\ (\boldsymbol{x_j}(u_0) = \boldsymbol{v_j}) \ \wedge\ \textstyle\bigwedge_{l=i+1}^{k} f_j^l = \mathtt{m}_{F_a(\boldsymbol{x_j},t)}.$$

Let $tr(\phi^T_{\mathcal{E}\restriction_{\Pi} E_{\mathcal{E}}})$ be the transformed formula obtained from the logical formula $\phi^T_{\mathcal{E}\restriction_{\Pi} E_{\mathcal{E}}}$ by repeatedly performing this procedure for every ODE subformula.

Definition 23. *For an ODE formula* $\forall t \in [u_0, u_t].\, \boldsymbol{x_j}(t) = \boldsymbol{v_j} + \int_0^{t-u_0} F(\boldsymbol{x_j}, t)\,\mathrm{d}t$ *for a subcomponent* $M_j \restriction_{\pi_j} E_{M_j}$*, if name constant* $\mathtt{m}_{F(\boldsymbol{x_j},t)}$ *denotes* $F(\boldsymbol{x_j}, t)$*, then its ODE-free transformation is the formula:*

$$\bigvee_{0\leq i\leq k\leq n} \left[(t_i = u_0) \wedge (t_k = u_t) \ \wedge\ (\boldsymbol{x_j}(u_0) = \boldsymbol{v_j}) \ \wedge\ \bigwedge_{l=i+1}^{k} f_j^l \equiv \mathtt{m}_{F(\boldsymbol{x_j},t)} \right].$$

Example 14. For the formula $\phi^{60}_{\mathcal{E}\restriction_{\Pi} E_{\mathcal{E}}}$ in Example 10 for the simple airplane controller, as explained in Example 13, the beginning and end times for interval assignments are given by the set $\{15i, 15i+u^L_{R,1}, 15i+u^R_{R,1} \mid i = 0,1,2,3\} \cup \{60\}$, and therefore $N = 12$. We define time variables $t_1, t_2, \ldots, t_{12}$, and name variables $f^1_{Main}, \ldots, f^{12}_{Main}, f^1_L, \ldots, f^{12}_L, f^1_R, \ldots, f^{12}_R$. For the main controller, the formula $\forall t \in [0, 60].\, [\tau_\phi, \tau_p](t) = [y_{v_\phi}, y_{v_p}] + \int_0^{t-y_{u_0}} [\tau_p(t), c(\tau_{\delta_R}(t) - \tau_{\delta_L}(t))]\mathrm{d}t$ becomes

$$[\tau_\phi, \tau_p](0) = [y_{v_\phi}, y_{v_p}] \ \wedge\ \bigwedge_{l=1}^{12} f^l_{Main} \equiv \mathtt{m}_{[\tau_p(t), c(\tau_{\delta_R}(t) - \tau_{\delta_L}(t))]}.$$

For each subcontroller M and its intermediate step $i = 0, 1, 2, 3$, the formulas $(\forall t \in [15i, 15i + u^M_R].\, \tau_M(t) = y^i_{v_M} + \int_0^{u^M_R} y^I_{rate^i_M}\,\mathrm{d}t)$ and $(\forall t \in [15i + u^M_R, 15(i+1)].\, \tau_M(t) = y^i_{v_M} + \int_0^{15-u^M_R} y^R_{rate^i_M}\,\mathrm{d}t)$, respectively, become:

$$\textstyle\bigvee_{k=3i+1}^{3i+2} \left(\tau_M(15i) = y^i_{v_M} \ \wedge\ \bigwedge_{l=3i+1}^{k} f^l_M = \mathtt{m}_{y^I_{rate^i_M}} \right)$$

$$\textstyle\bigvee_{k=3i+2}^{3(i+1)} \left(\tau_M(15i + u^M_R) = y^i_{v_M} \ \wedge\ \bigwedge_{l=k}^{3(i+1)} f^l_M = \mathtt{m}_{y^R_{rate^i_M}} \right)$$

Notice that we have already discarded unsatisfiable time segment assignments from these formulas (e.g., $t_0 = 30$ and $t_{12} = 30 + u_R^L$).

Finally, we build complete systems of ODEs *parameterized by name variables*, using time segments of a global period $[0, T]$. Consider a subinterval $[t_{i-1}, t_i]$ and its corresponding name variables $\{f_j^i\}_{j \in J_S \cup J_F}$. Suppose that $\{\boldsymbol{g}_{i-1}^j\}_{j \in J_S \cup J_F}$ denote the values of the physical parameters at time t_{i-1}. Then, the relationship between those variables are expressed by:

$$[\boldsymbol{g}_i^j]_{j \in J_S \cup J_F} = [\boldsymbol{g}_{i-1}^j]_{j \in J_S \cup J_F} + \int_0^{t_i - t_{i-1}} [f_j^i]_{j \in J_S \cup J_F} \, \mathrm{d}t,$$

written as a term $[\boldsymbol{g}_i^j]_{j \in J_S \cup J_F} = [\boldsymbol{g}_{i-1}^j]_{j \in J_S \cup J_F} + int^{\boldsymbol{l}}(t_i - t_{i-1}, [f_j^i]_{j \in J_S \cup J_F})$, where $\boldsymbol{l}$ denotes the dimensions of the physical environments.

Definition 24. *Given a number N, a* global ODE formula $\phi_{ODE}(\boldsymbol{f}, \boldsymbol{t}, \boldsymbol{g})$ *is a conjunction of N ODE formulas, parameterized by name variables:*

$$\boldsymbol{x}(t_0) = \boldsymbol{g}^0 \ \wedge \bigwedge_{1 \le i \le N} (\boldsymbol{g}^i = \boldsymbol{g}^{i-1} + int^{\boldsymbol{l}}(t_i - t_{i-1}, \boldsymbol{f}^i) \ \wedge \ \boldsymbol{x}(t_i) = \boldsymbol{g}^i),$$

where: (i) $\boldsymbol{t} = \{t_0, t_1, \ldots, t_N\}$ are time variables *denoting N time segments with $0 = t_0 \le t_1 \le t_2 \le \cdots \le t_N = T$; (ii) $\boldsymbol{f} = \{f_j^1, \ldots, f_j^N\}_{j \in J_S \cup J_F}$ are* name variables *denoting (partial) ODE systems of each E_{M_j} for each subinterval $[t_{i-1}, t_i]$; (iii) $\boldsymbol{g} = \{\boldsymbol{g}_j^0, \ldots, \boldsymbol{g}_j^N\}_{j \in J_S \cup J_F}$ are* value variables *denoting the values of all physical parameters of $\mathcal{E} \restriction_\Pi E_\mathcal{E}$ at each time t_i; and (iv) $\boldsymbol{l} = \{l_j\}_{j \in J_S \cup J_F}$ are the* dimensions *of E_{M_j}'s parameters $\boldsymbol{x_j} = (x_j^1, \ldots, x_j^{l_j})$.*

Example 15. Consider the time segments in Examples 13 and 14 for the simple airplane controller. The global ODE formula ϕ_{ODE} is given by:

$$[\tau_\phi(t_0), \tau_p(t_0), \tau_L(t_0), \tau_R(t_0)] = [g_\phi^0, g_p^0, g_L^0, g_R^0] \ \wedge$$

$$\bigwedge_{i=1}^{12} \left(\begin{bmatrix} g_\phi^i \\ g_p^i \\ g_L^i \\ g_R^i \end{bmatrix} = \begin{bmatrix} g_\phi^{i-1} \\ g_p^{i-1} \\ g_L^{i-1} \\ g_R^{i-1} \end{bmatrix} + int^{2,1,1}(t_i - t_{i-1}, \begin{bmatrix} f_{Main}^i \\ f_L^i \\ f_R^i \end{bmatrix}) \ \wedge \ \begin{bmatrix} \tau_\phi(t_i) \\ \tau_p(t_i) \\ \tau_L(t_i) \\ \tau_R(t_i) \end{bmatrix} = \begin{bmatrix} g_\phi^i \\ g_p^i \\ g_L^i \\ g_R^i \end{bmatrix} \right)$$

Using ODE-free-transformations and ϕ_{ODE}, we obtain the formula that includes no uninterpreted real functions and universal quantification:

$$\tilde{\phi}^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}} \equiv (\exists \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{g}).\ \phi_{ODE}(\boldsymbol{f}, \boldsymbol{t}, \boldsymbol{g}) \wedge tr(\phi^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}).$$

Theorem 4. *$\phi^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ is satisfiable iff $\tilde{\phi}^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ is satisfiable.*

Proof (Proof Sketch). ($\Rightarrow$) If the logical formula $\phi^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ is satisfiable, then there exists a collection $\{\boldsymbol{x_j}\}_{j \in J_S \cup J_F}$ of trajectories for a global round, and we can find the concrete values of $(\boldsymbol{f}, \boldsymbol{t}, \boldsymbol{g})$ from the trajectories. ($\Leftarrow$) If $\tilde{\phi}^T_{\mathcal{E} \restriction_\Pi E_\mathcal{E}}$ is satisfiable,

then there exist resulting values of $(\boldsymbol{f}, \boldsymbol{t}, \boldsymbol{g})$, and a collection $\{\boldsymbol{x_j}\}_{j \in J_S \cup J_F}$ of trajectories can be constructed. Then, by construction, any ODE subformula $\forall t \in [u_0, u_t]. \boldsymbol{x_j}(t) = \boldsymbol{v_j} + \int_0^{t-u_0} F(\boldsymbol{x_j}, t)\, \mathrm{d}t$ is true iff its transformed formula by Definition 23 is true, where $u_0 = t_i$ and $u_t = t_k$ for some $0 \leq i \leq k \leq n$. □

Notice that in the new encoding $\tilde{\phi}^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$ ODE literals (i.e., terms including integral operators) only appear in the global ODE subformula ϕ_{ODE}. Although it would be possible to represent Hybrid PALS models in the standard SMT encoding using the time segment approach, the size of such a formula is much bigger than the size of $\tilde{\phi}^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$. For each time segment, an old encoding can include many ODE literals to represent different combinations of complete ODE systems, but there is only one ODE literal in the modular encoding $\tilde{\phi}^T_{\mathcal{E}\restriction_\Pi E_\mathcal{E}}$, parameterized by name variables. Because the most computationally expensive operation is a decision procedure for ODEs, minimizing the number of ODE literals is a very effective way to enhance the performance of analysis.

5 Case Studies and Experimental Results

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`.

5.1 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. 8. 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}. In each round, M receives $goal_M$

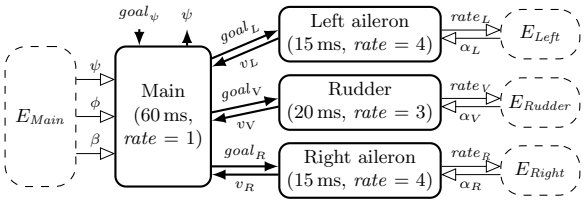


Fig. 8. The distributed controllers for turning an airplane.

from M_{Main}, determines the moving rate $rate_M$ based on $goal_M$ and the current sampled value v_M of the surface angle α_M, and sends back v_M to M_{Main}.[13] The continuous dynamics of α_M is specified by the local physical environment E_M as the ODE $\frac{\mathrm{d}\alpha_M}{\mathrm{d}t} = rate_M$,

The main controller M_{Main} determines the goal angles for the subcontrollers to make a coordinated turn. In each round, M_{Main} receives a desired direction $goal_\psi$ (from the pilot) and the surface angles (v_L, v_V, v_R) from the subcontrollers, and then sends back the new goal angles $(goal_L, goal_V, goal_R)$, based on the current sampled *position* values $(v_\psi, v_\phi, v_\beta)$ of the direction angle ψ, the roll angle ϕ, and the yaw angle β. We use a simple control logic to decide the new goal angles based on the current position angles, namely, by using some function $(goal_L, goal_V, goal_R) = f_{Main}(v_\psi, v_\phi, v_\beta)$.[14]

The local physical environment E_{Main} specifies the lateral dynamics of an aircraft as the nonlinear ODEs [19] (depicted in Fig. 9):

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

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 δ_M in E_{Main} must be the same as the corresponding surface angle α_M. Such immediate physical correlations between the local environments are specified by the time-invariant constraint

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

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

[14] For example, we can use three control functions [2]: (i) $f_\phi(v_\phi, v_\psi, goal_\psi)$ returns the goal roll angle $goal_\phi$, (ii) $f_R(v_\phi, goal_\phi)$ returns the goal angle $goal_R$ for the right aileron (where the goal angle $goal_L$ for the left aileron is the opposite angle $-goal_R$); and (iii) $f_V(v_\beta)$ returns the goal rudder angle $goal_V$. For example, for $h = 0.32(g_\psi - v_\psi)$, $f_\phi(v_\phi, v_\psi, g_\psi) = v_\phi + \mathrm{sign}(h - v_\phi)\cdot\min(\mathrm{abs}(h - v_\phi), 1.5)$, $f_R(v_\phi, g_\phi) = \mathrm{sign}(g_\phi - v_\phi)\cdot\min(0.3\,\mathrm{abs}(g_\phi - v_\phi), 15))$, and $f_V(v_\beta) = \mathrm{sign}(-v_\beta)\cdot\min(0.3\,\mathrm{abs}(v_\beta), 10)$.

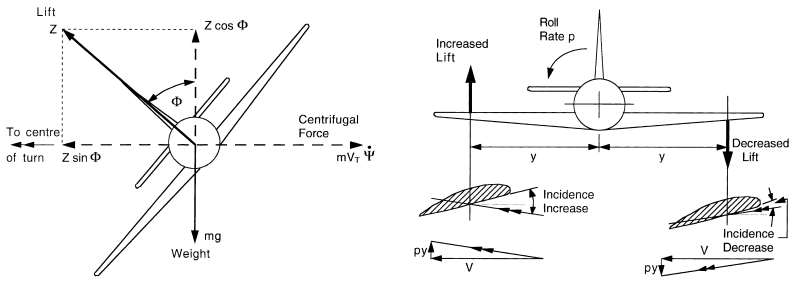


Fig. 9. Forces acting in a turn of an aircraft [7].

Notice that since the main controller and the subcontrollers have *different periods with local clock skews*, the ODEs of the subcontrollers cannot be directly "plugged" into E_{Main}. The PALS models consist of the multirate ensemble $\mathcal{E}$, the physical environments, and the time-invariant constraint, along with appropriate sampling and response timing policies Π.

We first performed bounded reachability analysis to verify the safety property $\forall t.\, \mathrm{abs}(\beta(t)) < 0.2$, 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 $t_I = 0\,\mathrm{ms}$ for every controller, the response time $t_R = 3\,\mathrm{ms}$ for every subcontroller (the main controller has no actuator), and the maximal clock skew $\epsilon = 0.2\,\mathrm{ms}$. For bound $k = 10$, the analysis for the entire system 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.[15] Next, assuming that a subcontroller cannot abruptly move its surface (that is, the output conditions of the subcontrollers become the input condition of the main controller), 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 $\forall t.\, \mathrm{abs}(\beta(t)) < 0.2$ took 2 minutes.

5.2 Networked Water Tank Controllers

In this benchmark, adapted from [15,18], a number of water tanks are connected by pipes as shown in Fig. 10. The water level in each tank is controlled by a pump in the tank, and depends on the pump's mode $m \in \{m_{\mathtt{on}}, m_{\mathtt{off}}\}$ and the water level of the input tank. The water level x_i of tank i changes according to

[15] E.g., the output condition for the left subcontroller is $c^L_{out}(v_L, \alpha_L(t)) \equiv \forall t \in [0,T].\, \mathrm{abs}(\alpha_L(t) - v_L) < \gamma$ for some $\gamma > 0$.

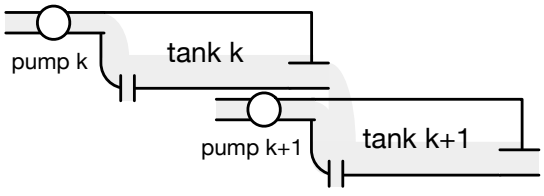


Fig. 10. Connected water tanks.

the nonlinear ODEs:

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

where A_i, q_i, a, b are constants determined by the size of the tank, the power of the pump, and the widths of the I/O pipes, and g is the standard gravity constant. 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 $x_i \leq L_m$ 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 $[0, T]$.[16] 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 $\forall t \in [0, T].\, c^{i-1}_{in}(x_{i-1}(t))$ implies $\forall t \in [0, T].\, x_i(t) \in I$) for a period T (the inductive condition and the safety condition are identical).

We have proved this compositional safety property for any number of water tanks for maximal clock skew $\epsilon = 30\,\text{ms}$, sampling time $t_I = 20\,\text{ms}$, and response time $t_R = 100\,\text{ms}$, by checking the unsatisfiability of the negated formulas, with precision $\delta = 0.001$ using dReal (the analysis took 4.3 seconds).[17] 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 by the trajectory in Fig. 11 generated by dReal with precision $\delta = 0.001$ (the analysis took 1.46 seconds).

5.3 Networked Thermostat Controllers

A number of rooms are interconnected by open doors, as shown in Fig. 12. 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

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

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

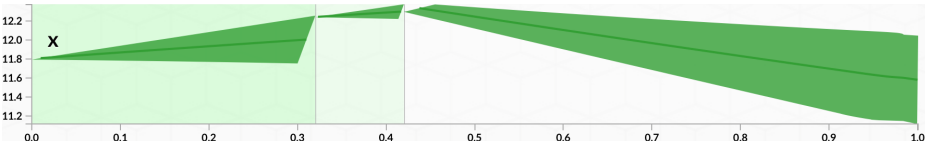


Fig. 11. The counterexample trajectory when $\epsilon = 150\,\text{ms}$, where the water level increases for extra 300 ms to violate the inductive condition.

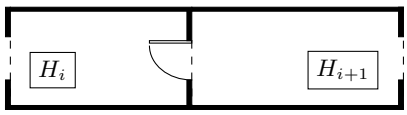


Fig. 12. Interconnected rooms.

heater's mode $m \in \{m_{\texttt{on}}, m_{\texttt{off}}\}$ and the temperatures of the connected rooms, and changes according to the ODEs:

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

where $K_i, h_i \in \mathbb{R}$ are constants depending on the size of room i and the heater's power, respectively, and $c \in \mathbb{R}$ is determined by the size of the open door. In each transition, a controller of room i turns on the heater if $x_i \leq T_m$, 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, [18] 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. [19] 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 5.2.

We have proved the safety property for any number of thermostats by checking the unsatisfiability of the negated formulas, 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). [20]

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 by the trajectory in Fig. 13 (the analysis took 0.56 seconds).

[18] $c^i_{in}(\langle x_{i-1}, x_{i+1}\rangle(t)) \equiv (\forall t \in [0,T].\, x_{i-1}(t), x_{i+1}(t) \in I)$.
[19] $c^i_{out}(x_i(t)) \equiv (\forall t \in [0,T].\, x_i(t) \in I')$.
[20] In the analysis, we use the parameters $K = 0.015$, $h = 100$, $c = 0.01$, $T_M = 21$, $T_m = 19$, $\eta = 3$, $\eta' = 2$, and $T = 1\,\text{s}$.

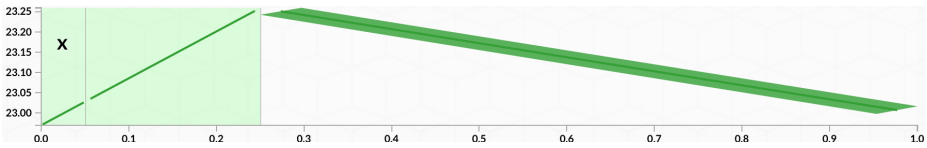


Fig. 13. The counterexample trajectory when $\epsilon = 20\,\text{ms}$, where the temperature rises for extra 20 ms and thus $x > 23$ at the end of the round.

6 Related Work.

PALS [1, 3, 17] 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, 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 [4]. However, the formal models of Hybrid PALS in [4] are very different from the models in this work, so that a bisimulation equivalence could not be provided in [4]. In this paper, Hybrid PALS models are significantly re-engineered 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 [4], but those times are synchronized in this paper to properly model the continuous behavior for *tightly coupled* environments. Furthermore, [4] 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., [6,11,14]), and their compositional analysis for any number of identical processes is related to [13]. 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.

7 Concluding Remarks

We have presented verification techniques fo 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.

References

1. Al-Nayeem, A., Sun, M., Qiu, X., Sha, L., Miller, S.P., Cofer, D.D.: A formal architecture pattern for real-time distributed systems. In: IEEE RTSS (2009)
2. Bae, K., Krisiloff, J., Meseguer, J., Ölveczky, P.C.: Designing and verifying distributed cyber-physical systems using Multirate PALS: An airplane turning control system case study. Sci. Comp. Prog. 103 (2015)
3. Bae, K., Meseguer, J., Ölveczky, P.C.: Formal patterns for multirate distributed real-time systems. Sci. Comp. Program. 91, 3–44 (2014)
4. Bae, K., Ölveczky, P.C.: Hybrid Multirate PALS. In: Logic, Rewriting, and Concurrency. LNCS, vol. 9200. Springer (2015)
5. Bae, K., Kong, S., Gao, S.: Smt encoding of hybrid systems in dreal. In: ARCH Workshop (2015), http://cps-vo.org/node/20310
6. Bogomolov, S., Herrera, C., Muñiz, M., Westphal, B., Podelski, A.: Quasi-dependent variables in hybrid automata. In: HSCC. ACM (2014)
7. Collinson, R.P.: Introduction to avionics systems. Springer (2013)
8. Gao, S., Avigad, J., Clarke, E.M.: δ-complete decision procedures for satisfiability over the reals. In: IJCAR. LNCS, vol. 7364. Springer (2012)
9. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: CADE. LNCS, vol. 7898. Springer (2013)
10. Gao, S., Kong, S., Clarke, E.M.: Satisfiability modulo ODEs. In: FMCAD. IEEE (2013)
11. Hendriks, M., Behrmann, G., Larsen, K.G., Niebert, P., Vaandrager, F.W.: Adding symmetry reduction to Uppaal. In: FORMATS. LNCS, vol. 2791 (2003)
12. Henzinger, T.A.: The theory of hybrid automata. Springer (2000)
13. Johnson, T.T., Mitra, S.: A small model theorem for rectangular hybrid automata networks. In: FMOODS/FORTE. LNCS 7273, Springer (2012)
14. Johnson, T.T., Mitra, S.: Anonymized reachability of hybrid automata networks. In: FORMATS, LNCS, vol. 8711. Springer (2014)

15. Kowalewski, S., Stursberg, O., Fritz, M., Graf, H., Hoffmann, I., Preußig, J., Remelhe, M., Simon, S., Treseler, H.: A case study in tool-aided analysis of discretely controlled continuous systems: The two tanks problem. In: Hybrid Systems V, LNCS, vol. 1567. Springer (1999)
16. Lynch, N., Segala, R., Vaandrager, F.: Hybrid I/O automata. Infor. and Comput. 185(1) (2003)
17. Meseguer, J., Ölveczky, P.C.: Formalization and correctness of the PALS architectural pattern for distributed real-time systems. Theoretical Computer Science 451, 1–37 (2012)
18. Raisch, J., Klein, E., O'Young, S., Meder, C., Itigin, A.: Approximating automata and discrete control for continuous systems. In: Hybrid Systems V, LNCS, vol. 1567. Springer (1999)
19. Stevens, B.L., Lewis, F.L.: Aircraft control and simulation. John Wiley & Sons (2003)

A Hybrid Automata Semantics

A hybrid automaton [12] is one of the most widely used formalism for modeling hybrid systems. This section explains the relationship between our Hybrid PALS formalism and hybrid automata. The hybrid automata representation of a Hybrid PALS synchronous model is quite complex due to extra timer variables and communication jump conditions as shown in this section.

A.1 Preliminaries on Hybrid Automata

A state of a hybrid automaton is a pair $(m, \boldsymbol{v})$ of a discrete mode m and a vector $\boldsymbol{v} \in \mathbb{R}^l$ of real numbers. The behavior of a hybrid automaton is specified by using both discrete *jump* conditions and continuous *flow* conditions. Let $\mathfrak{T}$ be a set of trajectories over a closed time interval beginning at 0, e.g., $f : [0, \infty) \to \mathbb{R}^l$ or $f : [0, t] \to \mathbb{R}^l$ for $t \geq 0$ (see [12, 16] for general conditions of trajectories).

Definition 25. *A* hybrid automaton $H = (X, Q, \Sigma, \mathit{init}, \mathit{inv}, \mathit{flow}, \mathit{jump})$ *has:*

- $X = \{x_1, \ldots, x_l\}$ *a finite set of real-numbered variables* ($\mathit{val}(X) = \mathbb{R}^l$)*;*
- Q *a set of control modes, and* Σ *a set of actions;*
- $\mathit{init} \subseteq Q \times \mathit{val}(X)$ *a set of initial states;*
- $\mathit{inv} : Q \to 2^{\mathit{val}(X)}$ *assigning to each mode* q *its invariant condition (i.e., a set of all possible values of* X *in mode* q*);*
- $\mathit{flow} : Q \to 2^{\mathfrak{T}}$ *assigning to each mode* q *its flow condition (i.e., a set of all trajectories of* X *in mode* q*); and*
- $\mathit{jump} : (Q \times \mathit{val}(X)) \times \Sigma \times (Q \times \mathit{val}(X))$ *a jump condition to define a discrete transition* $(q, \boldsymbol{v}) \xrightarrow{a} (q', \boldsymbol{v}')$ *with action* a*.*

Note that *init*, *inv*, and *flow* are often defined as predicates over the variables X: e.g., for each mode $q \in Q$, predicates: $\mathit{init}_q(x_1, \ldots, x_l)$, $\mathit{inv}_q(x_1, \ldots, x_l)$, and $\mathit{flow}_q(x_1, \ldots, x_l, \dot{x}_1, \ldots, \dot{x}_l)$, where $\dot{x}_i$ denotes the first derivative $\frac{\mathrm{d}x_i}{\mathrm{d}t}$ of x_i.

Now consider two hybrid automata H_1 and H_2. A (discrete) communication between H_1 and H_2 is modeled by joint synchronous actions in $\Sigma_1 \cap \Sigma_2$, and their (continuous) interaction is modeled by using shared variables in $X_1 \cap X_2$. Let $\boldsymbol{v} \restriction_{X_i} \in \mathit{val}(X_i)$, $i = 1, 2$, denote the restriction of $\boldsymbol{v} \in \mathit{val}(X_1 \cup X_2)$ by X_i in which both $\boldsymbol{v}$ and $\boldsymbol{v} \restriction_{X_i}$ give the same value for each common variable $x \in X_i$.

Definition 26. *Given two automata* $H_i = (X_i, Q_i, \Sigma_i, \mathit{init}_i, \mathit{inv}_i, \mathit{flow}_i, \mathit{jump}_i)$ *for* $i = 1, 2$*, their parallel composition* $H_1 \parallel H_2$ *is defined by the hybrid automata* $H_1 \parallel H_2 = (X_1 \cup X_2,\ Q_1 \times Q_2,\ \Sigma_1 \cup \Sigma_2,\ \mathit{init},\ \mathit{inv},\ \mathit{flow},\ \mathit{jump})$*, where:*

- $\mathit{init} = \{((q_1, q_2), \boldsymbol{v}) \mid (q_1, \boldsymbol{v} \restriction_{X_1}) \in \mathit{init}_1, (q_2, \boldsymbol{v} \restriction_{X_2}) \in \mathit{init}_2\}$*;*
- $\mathit{inv}(q_1, q_2) = \{\boldsymbol{v} \in \mathit{val}(X_1 \cup X_2) \mid \boldsymbol{v} \restriction_{X_1} \in \mathit{inv}_1(q_1), \boldsymbol{v} \restriction_{X_2} \in \mathit{inv}_2(q_2)\}$*;*
- $f \in \mathit{flow}$ *iff there exist compatible trajectories* $f_1 \in \mathit{flow}_i$*,* $i \in 1, 2$*, such that* $\mathit{dom}(f) = \mathit{dom}(f_i)$ *and* $(\forall t \in \mathit{dom}(f))\ f(t) \restriction_{X_i} = f_i(t)$*; and*
- $((q_1, q_2), \boldsymbol{v}) \xrightarrow{a} ((q_1', q_2'), \boldsymbol{v}') \in \mathit{jump}$ *iff* $a \in \Sigma_1 \cup \Sigma_2$ *and for each* $i = 1, 2$*:*

$$\begin{cases} (q_i, \boldsymbol{v} \restriction_{X_i}) \xrightarrow{a} (q_i', \boldsymbol{v}' \restriction_{X_i}) \in \mathit{jump}_i & \text{if } a \in \Sigma_i \\ (q_i, \boldsymbol{v} \restriction_{X_i}) = (q_i', \boldsymbol{v}' \restriction_{X_i}) & \text{if } a \notin \Sigma_i. \end{cases}$$

A.2 Hybrid Automata for Hybrid PALS Synchronous Models

We can define a hybrid automaton for a Hybrid PALS synchronous model $\mathcal{E}$. For each typed machine M and its physical environment $\mathcal{E}_M$ in a multirate ensemble $\mathcal{E}$, there exists a corresponding hybrid automaton $H_{M \upharpoonright E_M}$ in which the discrete jumps are given by the transitions of M and the continuous flows are given by the physical transitions of $\mathcal{E}_M$. Recall that a communication between components in $\mathcal{E}$ happens at the beginning of their hyper-period. Therefore, we extend each $H_{M \upharpoonright E_M}$ to the communication hybrid automaton $\bar{H}_{M_j}$ by adding discrete jumps for such synchronous communications, and then their composition $\|_{M_j \in \mathcal{E}} \bar{H}_{M_j}$ becomes the hybrid automaton for $\mathcal{E}$. Finally, the time-invariant constraints for $\mathcal{E}$ are encoded in $\|_{M_j \in \mathcal{E}} \bar{H}_{M_j}$ using shared variables. For simplicity reasons, in this section we assume that sampling times are always 0 and response times are maximal execution times α_M.

The environment restriction $M \upharpoonright E_M$ of a controller $M = (D_i, S, D_o, \delta_M)$ by its environment $E_M = (C, \boldsymbol{x}, U, \Lambda)$ corresponds to a hybrid automaton $H_{M \upharpoonright E_M}$. The variables of $H_{M \upharpoonright E_M}$ include the physical parameters $\boldsymbol{x}$ of E_M, and an extra *timer* variable τ_M to keep track of its periods. A discrete state of $H_{M \upharpoonright E_M}$ consists of a state of M, input buffers $D_{i\perp}$, and output buffers $D_{o\perp}$, where

$$D_{i\perp} = (D_{i_1} \cup \{\perp\}) \times \cdots \times (D_{i_n} \cup \{\perp\}),$$
$$D_{o\perp} = (D_{o_1} \cup \{\perp\}) \times \cdots \times (D_{o_m} \cup \{\perp\}),$$

with $\perp$ an empty buffer. As $M \upharpoonright E_M$, the observed behavior $\pi_{\boldsymbol{x}}(s)$ by M in a state $s \in S$ should be the same as the observable behavior $\pi_S(\boldsymbol{v})$ of E_M. Initially, the timer τ_M is set to 0. During each period when $0 \leq \tau_M \leq c_M(\pi_t(s')) - c_M(\pi_t(s))$, the value of the parameter $\boldsymbol{x}$ changes according to E_M. At the end of each period (when $\tau_M = c_M(\pi_t(s')) - c_M(\pi_t(s))$), a discrete transition $((\boldsymbol{i}, s), (s', \boldsymbol{o})) \in \delta_M$ is taken using the value $\boldsymbol{i}$ in its input buffer, the output $\boldsymbol{o}$ is stored in its output buffer, and the timer τ_M is reset to 0. The value in an input buffer is updated in the ensemble level (see below).

Definition 27. *Given a machine $M = (D_i, S, D_o, \delta_M)$, its physical environment $E_M = (C, \boldsymbol{x}, U, \Lambda)$ with $\boldsymbol{x} = (x_1, \ldots, x_l)$, and a local clock $c_M : \mathbb{N} \to \mathbb{R}_{>0}$, the corresponding automaton is $H_{M \upharpoonright E_M} = (X, Q, \Sigma, init, inv, flow, jump)$, where:*

- $X = \{x_1, \ldots, x_l, \tau_M\}$ *with τ_M a δ_M-timer variable;*
- $Q = S \times D_{i\perp} \times D_{o\perp}$ *with $D_{i\perp}$ input buffers and $D_{o\perp}$ output buffers;*
- $\Sigma = \{\vartheta_M\}$ *with ϑ_M an* internal *action;*
- $init = \{((s, \boldsymbol{i}, \perp), (\boldsymbol{v}, 0)) \in Q \times \mathbb{R}^{l+1} \mid \pi_t(s) = 0, \pi_{\boldsymbol{x}}(s) = \pi_S(\boldsymbol{v})\}$;
- $inv(s, \boldsymbol{i}, \boldsymbol{o}) = \{(\boldsymbol{v}, t) \in \mathbb{R}^{l+1} \mid 0 \leq t \leq c_M(\pi_t(s) + 1) - c_M(\pi_t(s))\}$;
- $(\boldsymbol{f}, f_{\tau_M}) \in flow(s, \boldsymbol{i}, \boldsymbol{o})$ *iff* $\dot{f}_{\tau_M} = 1$ *and* $(\exists \boldsymbol{v} \in \mathbb{R}^l)\ ((\pi_C(s), \boldsymbol{v}), \boldsymbol{f}) \in \Lambda$; *and*
- $((s, \boldsymbol{i}, \perp), (\boldsymbol{v}', T)) \xrightarrow{\vartheta_M} ((s', \perp, \boldsymbol{o}), (\boldsymbol{v}', 0)) \in jump$ *iff:*

$$((\boldsymbol{i}, s), (s', \boldsymbol{o})) \in \delta_M, \quad \pi_{\boldsymbol{x}}(s') = \pi_S(\boldsymbol{v}'), \quad T = c_M(\pi_t(s')) - c_M(\pi_t(s)).$$

The following lemma states that the behaviors of $M \upharpoonright E_M$ and $H_{M \upharpoonright E_M}$ are equivalent with respect to their periods.

Lemma 1. *Given an environment restriction $M \restriction E_M$ and its hybrid automaton $H_{M\restriction E_M}$, an $M \restriction E_M$'s transition $((\boldsymbol{i},(s,\boldsymbol{v})),((s',\boldsymbol{v}'),\boldsymbol{o})) \in \delta_{M\restriction E_M}$ holds iff:*

- $((s,\boldsymbol{i},\bot),(\boldsymbol{v},0)) \xrightarrow{T} ((s,\boldsymbol{i},\bot),(\boldsymbol{v}',T))$ *holds by flow of $H_{M\restriction E_M}$ during time $T = c_M(\pi_t(s)+1) - c_M(\pi_t(s))$ for $\pi_{\boldsymbol{x}}(s) = \pi_S(\boldsymbol{v})$; and*
- $((s,\boldsymbol{i},\bot),(\boldsymbol{v}',T)) \xrightarrow{\Im_M} ((s',\bot,\boldsymbol{o}),(\boldsymbol{v}',0))$ *holds by jump of $H_{M\restriction E_M}$.*

Proof. If $((\boldsymbol{i},(s,\boldsymbol{v})),((s',\boldsymbol{v}'),\boldsymbol{o})) \in \delta_{M\restriction E_M}$, then we have: $((\boldsymbol{i},s),(s',\boldsymbol{o})) \in \delta_M$, $((\pi_C(s),\boldsymbol{v}),\boldsymbol{f}) \in \Lambda$, $\boldsymbol{f}(0) = \boldsymbol{v}$, $\boldsymbol{f}(c_M(\pi_t(s')) - c_M(\pi_t(s))) = \boldsymbol{v}'$ $\pi_t(s') = \pi_t(s)+1$, $\pi_{\boldsymbol{x}}(s) = \pi_S(\boldsymbol{v})$, and $\pi_{\boldsymbol{x}}(s') = \pi_S(\boldsymbol{v}')$. First, by *jump* of $H_{M\restriction E_M}$, for the timer value $T = c_M(\pi_t(s)+1) - c_M(\pi_t(s)) = c_M(\pi_t(s')) - c_M(\pi_t(s))$, the transition $((s,\boldsymbol{i},\bot),(\boldsymbol{v}',T)) \xrightarrow{\Im_M} ((s',\bot,\boldsymbol{o}),(\boldsymbol{v}',0))$ clearly holds. Second, during time T, by *flow* and *inv* of $H_{M\restriction E_M}$, the value of $\boldsymbol{x}$ can follow the trajectory $\boldsymbol{f}$ from $\boldsymbol{v}$ and the value of the timer τ_M follows from 0 the differential equation $\dot{f}_{\tau_M} = 1$. Therefore, $((s,\boldsymbol{i},\bot),(\boldsymbol{v},0)) \xrightarrow{T} ((s,\boldsymbol{i},\bot),(\boldsymbol{v}',T))$ holds. The other direction is also similarly straightforward by the definitions of $M \restriction E_M$ and $H_{M\restriction E_M}$. □

Now consider an ensemble $\mathcal{E} = (J_S \cup J_F, e, \{M_j\}_{j\in J_S\cup J_F}, E, \mathit{src}, \mathit{rate}, \mathit{adap})$, where each M_j is given as an environment restriction. For each M_j, we have the corresponding hybrid automaton H_{M_j}. Each automaton H_{M_j} is extended into its communication hybrid automaton $\bar{H}_{M_j}$ with extra output buffers and communication jump conditions. If M_j is a slow machine with $\mathit{rate}(j) = 1$, then the output buffers are just $D_{o\bot}$. If M_j is a fast machine with $\mathit{rate}(j) > 1$, then M_j should perform $\mathit{rate}(j)$ internal transitions in a single step. Therefore, the extended automaton $\bar{H}_{M_j}$ should have $\mathit{rate}(j)$ output buffers for each output port: that is, $D_{o\bot}^{\mathit{rate}(j)} = (D_{o_1} \cup \{\bot\})^{\mathit{rate}(j)} \times \cdots \times (D_{o_m} \cup \{\bot\})^{\mathit{rate}(j)}$. For output buffers $(\boldsymbol{o}_1,\ldots,\boldsymbol{o}_m) \in D_{o\bot}^{k}$ with k items, the append operation is given by $(\boldsymbol{o}_1,\ldots,\boldsymbol{o}_m);(d_1,\ldots,d_m) = ((\boldsymbol{o_1},d_1),\ldots,(\boldsymbol{o}_m,d_m)) \in D_{o\bot}^{k+1}$.

When the h-th input port (j,h) of M_j is connected to the l-th output port (k,l) of M_k in $\mathcal{E}$ (i.e., $\mathit{src}(j,h) = (k,l)$), their synchronous communication is expressed in hybrid automata using discrete jumps with action $\mathit{out}_{(k,l)}(d)$. There are two kinds of communication jumps: the "output" automaton $\bar{H}_{M_k}$ performs a discrete jump with action $\mathit{out}_{(k,l)}(d)$ when the value in its l-th output buffer is d, while all the "input" automata for each (j,h) such that $\mathit{src}(j,h) = (k,l)$ (recall that an output port can be connected to many input ports) perform jumps with action $\mathit{out}_{(k,l)}(d)$ that update the values in their input buffers by the new value d, together with their input adaptors $\mathit{adap}(j)$.

Such communication jumps must happen after every machine jump is taken, since machine jumps of H_{M_j} with period T_j happen at the *end* of its period, at each time $c_{M_j}(i)$ with $c_{M_j} = 0$ for $i = 0$ and $iT_j - \epsilon < c_{M_j}(i) < iT_j + \epsilon$ for $i > 0$, where $\epsilon \geq 0$ denotes a maximal clock skew, Therefore, each $\bar{H}_{M_j}$ has an extra timer variable τ_{src}, initially set to $T_C = T_{\mathcal{E}} + \epsilon$ with the hyper-period $T_{\mathcal{E}}$ (which is always equal to $T_j \cdot \mathit{rate}(j)$ for any $M_j \in \mathcal{E}$). The communication jumps are taken when $\tau_{\mathit{src}} = T_C$. Then, for the first synchronous step, τ_{src} is reset to 0 (so that next communication jumps happen after $T_{\mathcal{E}} + \epsilon$ elapsed), and for any other step, τ_{src} is reset to ϵ (so that next jumps happen after $T_{\mathcal{E}}$ elapsed).

Definition 28. *Consider a controller M_j with period T_j, its rate $rate(j)$, an input adaptor $adap(j) = \{\alpha_i\}_{l\in\{1,\dots,n\}}$, a wiring diagram src, and a maximal clock skew $\epsilon \geq 0$. Let $T_C = T_j \cdot rate(j) + \epsilon$. Given the machine hybrid automata $H_{M_j} = (X, Q, \Sigma, init, inv, flow, jump)$, the communication hybrid automata is defined by $\bar{H}_{M_j} = (\bar{X}, \bar{Q}, \bar{\Sigma}, \overline{init}, \overline{inv}, \overline{flow}, \overline{jump})$, where:*

- $\bar{X} = X \cup \{\tau_{src}\}$ *with τ_{src} a communication timer variable;*
- $\bar{Q} = S \times D_{i\perp} \times D_{o\perp}^{rate(j)}$, *if $Q = S \times D_{i\perp} \times D_{o\perp}$;*
- $\bar{\Sigma}$ *contains an out action for each connection in src involving M, that is:*

$$\begin{aligned}\bar{\Sigma} = \Sigma \cup \{out_{(k,l)}(d) \mid d \in D_{i_h}, src(j,h) = (k,l)\}\\ \cup \{out_{(j,h)}(d) \mid d \in D_{o_h}, src(k,l) = (j,h)\};\end{aligned}$$

- $\overline{init} = \{((s, \perp, \boldsymbol{o}), ((\boldsymbol{v}, 0), T_C)) \mid \pi_t(s) = 0, \pi_{\boldsymbol{x}}(s) = \pi_S(\boldsymbol{v})\}$ *(unlike init, input buffers are empty and output buffers are full);*
- $\overline{inv}(q) = \{(\boldsymbol{v}, t) \mid \boldsymbol{v} \in inv, 0 \leq t \leq T_C\}$ *(i.e., $0 \leq \tau_{src} \leq T_C$);*
- $(\boldsymbol{f}, f_{\tau_{src}}) \in \overline{flow}(q)$ *iff $\boldsymbol{f} \in flow(q)$ and $\dot{f}_{\tau_{src}} = 1$; and*
- $\overline{jump}$ *extends jump by adding extra communication jumps that, when τ_{src} is T_C: (i) receives inputs if input buffers are not full, (ii) sends outputs if output buffers are not empty, and (iii) resets the timer τ_{src} if input buffers are full and output buffers are empty. When $\tau_{src} < T_C$, $\overline{jump}$ performs the original jump and appends the output to the output buffer. That is:*
  - $(q, (\boldsymbol{v}, T_C)) \xrightarrow{out_{(k,l)}(d)} (rec_h(d,q), (\boldsymbol{v}, T_C)) \in \overline{jump}$, *if $src(j,h) = (k,l)$ and the h-th input buffer is empty, where $q = (s, (d_1, \dots, \perp, \dots, d_n), \boldsymbol{o})$ and $rec_h(d,q) = (s, (d_1, \dots, \alpha_h(d), \dots, d_n), \boldsymbol{o})$;*
  - $(q, (\boldsymbol{v}, T_C)) \xrightarrow{out_{(j,h)}(d)} (snd_h(d,q), (\boldsymbol{v}, T_C)) \in \overline{jump}$, *if $src(k,l) = (j,h)$ and the h-th output buffer is d, where $q = (s, \boldsymbol{i}, (d_1, \dots, d, \dots, d_m)))$ and $snd_h(d,q) = (s, \boldsymbol{i}, (d_1, \dots, \perp, \dots, d_m))$;*
  - $(q, (\boldsymbol{v}, T_C)) \xrightarrow{\jmath_M} (q, (\boldsymbol{v}, T_0)) \in \overline{jump}$, *if all the input buffers are full and all the output buffers are empty (i.e., $q = (s, \boldsymbol{i}, \perp)$ for $\boldsymbol{i} \in D_i$), where $T_0 = 0$ for the first synchronous step, and $T_0 = \epsilon$ for the other steps.*
  - $((s, \boldsymbol{i}, \boldsymbol{o}), (\boldsymbol{v}, t)) \xrightarrow{\jmath_M} ((s', \perp, \boldsymbol{o}; \boldsymbol{d}), (\boldsymbol{v}', t)) \in \overline{jump}$, *if $t < T_C$ holds and $((s, \boldsymbol{i}, \perp), \boldsymbol{v}) \xrightarrow{\jmath_M} ((s', \perp, \boldsymbol{d}), \boldsymbol{v}') \in jump$ (i.e., performing an original jump when $\tau_{src} < T_C$, and adding the output $\boldsymbol{d}$ to the output buffer $\boldsymbol{o}$).*

The composition $\|_{j\in J_S \cup J_F} \bar{H}_{M_j}$ becomes the desired hybrid automaton $H_{\mathcal{E}}$ for $\mathcal{E}$. Notice that an "output" action $out_{(j,h)}(d)$ of $\bar{H}_{M_j}$ can be taken only if there exists the same output action by other automata receiving d.[21] The time-invariant constraints for $\mathcal{E}$ can be easily encoded in each $\bar{H}_{M_j}$ by using shared variables and the invariant condition *inv* of hybrid automata (e.g., see Section A.3).

[21] This also implies that a self-loop connection for a machine M cannot be expressed by the above transformation using actions, because joint synchronous jumps for a single automaton is not allowed in hybrid automata. However, it is always possible to remove such self-loop connections using state variables.

Recall that each state of $\mathcal{E}$'s synchronous composition consists of the states $\{s_j, \boldsymbol{v}_j\}_{j\in J_S\cup J_F}$ of the subcomponents and the feedback outputs $\{\boldsymbol{d}_j\}_{j\in J_S\cup J_F}$ of the subcomponents to be transferred in the next step. For an ensemble state $(\{s_j, \boldsymbol{v}_j\}_{j\in J_S\cup J_F}, \{\boldsymbol{d}_j\}_{j\in J_S\cup J_F})$ of $\mathcal{E}$, its corresponding hybrid automaton state has the form $\|_{j\in J_S\cup J_F}((s_j, \boldsymbol{i}_j, \boldsymbol{o}_j), ((\boldsymbol{v}_j, t_j), T_C))$ for $T_C = T_{\mathcal{E}} + \epsilon$ with $T_{\mathcal{E}}$ the hyper-period, where each machine timer t_j (of τ_{M_j}) must be *consistent* with the communication timer t_{src} (of τ_{src}). The communication timer τ_{src} is initially set to T_C, and then immediately set to 0 after applying communication jumps. Whenever $\tau_{src} = T_C$ again, communication jumps are taken and τ_{src} is set to ϵ. That is, when $\tau_{src} = T_C$, the global time is either 0 or $nT_{\mathcal{E}} + \epsilon$ for a certain $n \in \mathbb{N}$. If the global time is 0, then $t_j = 0$. If the global time is $nT_{\mathcal{E}} + \epsilon$, which is in the $(n \cdot rate(j))$-th round for M_j, then $t_j = nT_{\mathcal{E}} + \epsilon - c_{M_j}(n \cdot rate(j))$.

Definition 29. *Given a hybrid state* $\|_{j\in J_S\cup J_F}((s_j, \boldsymbol{i}_j, \boldsymbol{o}_j), ((\boldsymbol{v}_j, t_j), T_C))$ *with* $T_C = T_{\mathcal{E}} + \epsilon$*, a machine timer* t_j *is* consistent *with* T_C *iff for the global time* T_g*:*

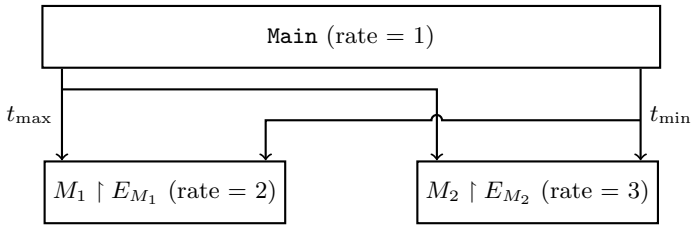
$$t_j = \begin{cases} 0 & \text{if } \ T_g = 0 \\ nT_{\mathcal{E}} + \epsilon - c_{M_j}(n \cdot rate(j)) & \text{if } \ (\exists n \in \mathbb{N}) \ \ T_g = nT_{\mathcal{E}} + \epsilon. \end{cases}$$

An ensemble $\mathcal{E}$ may also have external interface ports. Let us consider a *closed* ensemble with no interface ports The following lemmas show that the multirate synchronous composition of a closed ensemble $\mathcal{E}$ is equivalent to the composed hybrid automaton $\|_{j\in J_S\cup J_F} \bar{H}_{M_j}$ (the case of an *open* ensemble $\mathcal{E}$ with interface ports is similar, plus more complexity due to interface connections).

Lemma 2. *Given a closed ensemble* $\mathcal{E}$ *and its corresponding hybrid automaton* $H_{\mathcal{E}} = \|_{j\in J} \bar{H}_{M_j}$ *for* $J = J_S \cup J_F$*, assuming* $\epsilon < T_j$ *for any period* T_j *in* $\mathcal{E}$*, a transition* $((*, (\{s_j, \boldsymbol{v}_j\}_{j\in J}, \{\boldsymbol{d}_j\}_{j\in J})), ((\{s'_j, \boldsymbol{v}'_j\}_{j\in J}, \{\boldsymbol{d}'_j\}_{j\in J}), *)) \in \delta_{\mathcal{E}}$ *holds iff:*

- $\|_{j\in J}((s_j, \bot, \boldsymbol{d}_j), ((\boldsymbol{v}_j, t_j), T_C)) \longrightarrow^* \|_{j\in J}((s_j, \boldsymbol{i}_j, \bot), ((\boldsymbol{v}_j, t_j), T_0))$ *by only communication jumps, where* $T_0 = 0$ *for the first synchronous step,* $T_0 = \epsilon$ *for the other steps, and each timer* t_j *is consistent with* $T_{\mathcal{E}}$*, and*
- $\|_{j\in J}((s_j, \boldsymbol{i}_j, \bot), ((\boldsymbol{v}_j, t_j), T_0)) \longrightarrow^* \|_{j\in J}((s'_j, \bot, \boldsymbol{d}'_j), ((\boldsymbol{v}'_j, t'_j), T_C))$ *by using only machine jumps and flows with consistent timers* t_j*.*

Proof. Because $\mathcal{E}$ is closed, every output port in $\mathcal{E}$ is a feedback output port, and each ensemble state $(\{s_j, \boldsymbol{v}_j\}_{j\in J}, \{\boldsymbol{d}_j\}_{j\in J})$ corresponds to a hybrid automaton state $\|_{j\in J}((s_j, \bot, \boldsymbol{d}_j), ((\boldsymbol{v}_j, t_j), T_C))$. Because $\tau_{src} = T_C$, only communication jumps can be applied until all the input buffers are full and all the output buffers are empty. Then, τ_{src} is reset to T_0, either 0 (for the first step) or ϵ (for the other steps). In the resulting state $\|_{j\in J}((s_j, \boldsymbol{i}_j, \bot), ((\boldsymbol{v}_j, t_j), T_0))$, because of the timer consistency, all the machine transitions in the previous synchronous step must be taken, and any machine transition in the next step has *not* been taken yet, provided that $\epsilon < T_j$ for any period T_j. Now because $\tau_{src} < T_C$, only flows and machine jumps can be applied until $\tau_{src} = T_C$ and all the input buffers are empty again, resulting in the state $\|_{j\in J}((s'_j, \bot, \boldsymbol{d}'_j), ((\boldsymbol{v}'_j, t'_j), T_C))$. The other direction is straightforward by construction. □



$$(\forall t)\ x_{o_1}(t) = x_2(t)\ \wedge\ x_{o_2}(t) = x_1(t).$$

Fig. 14. A multirate ensemble $\mathcal{E}$.

A.3 Example

Figure 14 shows a multirate ensemble $\mathcal{E}$ for controlling the temperatures of the two adjacent rooms, one instance of physically networked thermostat controllers in Section 5.3. The ensemble $\mathcal{E}$ consists of three discrete components. The main controller `Main` sets a maximum temperature $t_{\max}$ and a minimum temperature $t_{\min}$ for both of the rooms, and each thermostat controller M_i $(i = 1, 2)$ then controls the room's heather according to given $t_{\max}$ and $t_{\min}$. The thermostat controllers M_1 and M_2 have different rates

$$\mathit{rate}(1) = 2 \quad \text{and} \quad \mathit{rate}(2) = 3.$$

That is, $2T_1 = 3T_2$ holds for the periods T_1 and T_2 of M_1 and M_2, respectively. Each thermostat controller in $\mathcal{E}$ is the environment restriction $M_i \upharpoonright E_{M_i}$, which also satisfies the also time-invariant constraint. Notice that the behavior of each $M_i \upharpoonright E_{M_i}$ is also parameterized by the exact local clock c_{M_i} of each controller M_i, since M_i "observes" the current temperature based on c_{M_i}.

This system can be expressed as the hybrid automaton $H_{M \upharpoonright E_M} = (X, Q, \{\mathfrak{I}_M\}, \mathit{init}, \mathit{inv}, \mathit{flow}, \mathit{jump})$, where[22]

- $X = \{x, x_o, \tau_M\}$ for the room temperature x, the outside temperature x_o, and the timer τ_M;
- $Q = \{m_{\mathtt{on}}, m_{\mathtt{off}}\} \times \mathbb{N} \times \mathbb{R}^2_\perp$, denoting a tuple $(m, n, t_{\max}, t_{\min})$ of mode m, local clock n, and input buffers $t_{\max}$ and $t_{\min}$;
- $\mathit{init} = \{((m, 0, t_M, t_m), (v, v_o, 0)) \mid t_M, t_m, v, v_o \in \mathbb{R}, m = \{m_{\mathtt{on}}, m_{\mathtt{off}}\}\}$;
- $\mathit{inv}(m, n, t_M, t_m) = \{(v, v_o, t) \in \mathbb{R}^3 \mid 0 \le t \le c_M(n+1) - c_M(n)\}$;
- $(f, f_o, f_{\tau_M}) \in \mathit{flow}(m, n, t_M, t_m)$ iff

$$\frac{\mathrm{d}f_{\tau_M}}{\mathrm{d}t} = 1, \qquad \frac{\mathrm{d}f}{\mathrm{d}t} = \begin{cases} K(h - ((1-k)f + kf_o)) & \text{if } m = m_{\mathtt{on}} \\ -K((1-k)f + kf_o) & \text{if } m = m_{\mathtt{off}}, \end{cases}$$

  where the constants $K, h, k \in \mathbb{R}$ depend on the size of the room, the power of the heater, and the size of the door, respectively;

[22] According to the original definition, $Q = \{m_{\mathtt{on}}, m_{\mathtt{off}}\} \times \mathbb{N} \times \mathbb{R} \times \mathbb{R}^3_\perp$ with extra observed parameters, which is omitted here for simplicity reasons, since observed parameters are always equal to observable parameter in $\mathit{val}(X)$.

– $((m, n, t_M, t_m), (v, v_0, T)) \xrightarrow{\jmath_M} ((m', n+1, \bot, \bot), (v, v_0, 0)) \in \mathit{jump}$ holds iff $T = c_M(n+1) - c_M(n)$, $m' = \textbf{if } v \leq t_{\max} \textbf{ then } m_{\texttt{on}} \textbf{ else } m_{\texttt{off}} \textbf{ fi}$ for $m = m_{\texttt{on}}$, and $m' = \textbf{if } v \geq t_{\min} \textbf{ then } m_{\texttt{off}} \textbf{ else } m_{\texttt{on}} \textbf{ fi}$ for $m = m_{\texttt{off}}$.

Now consider the multirate ensemble $\mathcal{E}$ in Figure 12 that consists of one main controller `Main` and two thermostat controllers M_1 and M_2 for two adjacent rooms, where $\mathit{rate}(\texttt{Main}) = 1$, $\mathit{rate}(1) = 2$, and $\mathit{rate}(2) = 3$. For T_i a period of M_i, $i = 1, 2$, let $T_C = 2 \cdot T_1 + \epsilon = 2 \cdot T_3 + \epsilon$. The communication automaton for M_1 is $H_{M_1 \uparrow E_{M_1}} = (\bar{X}_1, \bar{Q}_1, \bar{\Sigma}_1, \overline{\mathit{init}}_1, \overline{\mathit{inv}}_1, \overline{\mathit{flow}}_1, \overline{\mathit{jump}}_1)$, where:

– $\bar{X}_1 = \{x_1, x_{o_1}, \tau_{M_1}, \tau_{\mathit{src}}\}$, and $\bar{Q}_1 = \{m_{\texttt{on}}, m_{\texttt{off}}\} \times \mathbb{N} \times \mathbb{R}^2_\bot$ (no output port);
– $\bar{\Sigma}_1 = \{\jmath_{M_1}\} \cup \{\mathit{out}_{(\texttt{Main}, t_{\max})}(d) \mid d \in \mathbb{R}\} \cup \{\mathit{out}_{(\texttt{Main}, t_{\min})}(d) \mid d \in \mathbb{R}\}$;
– $\overline{\mathit{init}}_1 = \{(q, (v_1, v_{o_1}, 0, T_C)) \mid (q, (v_1, v_{o_1}, 0)) \in \mathit{init}_1\}$;
– $\overline{\mathit{inv}}_1(q) = \{(v_1, v_{o_1}, t, u) \mid (v_1, v_{o_1}, t) \in \mathit{inv}_1(q), 0 \leq u \leq T_C\}$;
– $(f_1, f_{o_1}, f_{\tau_M}, f_{\tau_{\mathit{src}}}) \in \overline{\mathit{flow}}(q)$ iff $(f_1, f_{o_1}, f_{\tau_M}) \in \mathit{flow}(q)$ and $\dot{f}_{\tau_{\mathit{src}}} = 1$; and
– When $\tau_{\mathit{src}} = T_C$, the communication jumps:

$$((m, n, \bot, t_m), (\boldsymbol{v}, T_C)) \xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\max})}(d_1, d_2)} ((m, n, d_2, t_m), (\boldsymbol{v}, T_C)) \in \overline{\mathit{jump}}_1,$$
$$((m, n, t_M, \bot), (\boldsymbol{v}, T_C)) \xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\min})}(d_1, d_2)} ((m, n, t_M, d_2), (\boldsymbol{v}, T_C)) \in \overline{\mathit{jump}}_1,$$
$$((m, n, t_M, t_m), (\boldsymbol{v}, T_C)) \xrightarrow{\jmath_M} ((m, n, t_M, t_m), (\boldsymbol{v}, T_0)) \in \overline{\mathit{jump}}_1 \quad (\text{if } t_M, t_m \in \mathbb{R}),$$

happen, where $T_0 = 0$ for the first step, and $T_0 = \epsilon$ for the other steps, and when $\tau_{\mathit{src}} < T_C$, the machine jumps happen (where $u < T_C$):

$$\begin{aligned} & ((m, n, t_M, t_m), (\boldsymbol{v}, u)) \xrightarrow{\jmath_M} ((m', n', \bot, \bot), (\boldsymbol{v}', u)) \in \overline{\mathit{jump}}_1 \\ \iff \; & ((m, n, t_M, t_m), \boldsymbol{v}) \xrightarrow{\jmath_M} ((m', n', \bot, \bot), \boldsymbol{v}') \in \mathit{jump}_1. \end{aligned}$$

The communication automaton $H_{M_2 \uparrow E_{M_2}}$ for M_2 is similar. The communication automaton for `Main` is $\bar{H}_{\texttt{Main}} = (\bar{X}, \bar{Q}, \bar{\Sigma}, \overline{\mathit{init}}, \overline{\mathit{inv}}, \overline{\mathit{flow}}, \overline{\mathit{jump}})$—which contains only timer variables, since the main controller `Main` is a discrete controller with no physical environments—where:

– $\bar{X} = \{\tau_{\texttt{Main}}, \tau_{\mathit{src}}\}$;
– $Q = \mathbb{N} \times \mathbb{R}^2_\bot$ for a tuple $(n, t_{\max}, t_{\min})$ of local clock n and output buffers;
– $\bar{\Sigma} = \{\jmath_{\texttt{Main}}\} \cup \{\mathit{out}_{(\texttt{Main}, t_{\max})}(d) \mid d \in \mathbb{R}\} \cup \{\mathit{out}_{(\texttt{Main}, t_{\min})}(d) \mid d \in \mathbb{R}\}$;
– $\overline{\mathit{init}} = \{((0, t_M, t_m), (0, T_C)) \mid t_M, t_m \in \mathbb{R}\}$;
– $\overline{\mathit{inv}}(n, t_M, t_m) = \{(t, u) \in \mathbb{R}^2 \mid 0 \leq t \leq c_{\texttt{Main}}(n+1) - c_{\texttt{Main}}(n), 0 \leq u \leq T_C\}$;
– $(f_{\tau_{\texttt{Main}}}, f_{\tau_{\mathit{src}}}) \in \overline{\mathit{flow}}(n, t_M, t_m)$ iff $\dot{f}_{\tau_{\texttt{Main}}} = 1$ and $\dot{f}_{\tau_{\mathit{src}}} = 1$;
– When $\tau_{\mathit{src}} = T_C$, the communication jumps:

$$((n, t_M, t_m), (\boldsymbol{v}, T_C)) \xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\max})}(t_M)} ((n, \bot, t_m), (\boldsymbol{v}, T_C)) \in \overline{\mathit{jump}} \quad (\text{if } t_M \in \mathbb{R}),$$
$$((n, t_M, t_m), (\boldsymbol{v}, T_C)) \xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\min})}(t_m)} ((n, t_M, \bot), (\boldsymbol{v}, T_C)) \in \overline{\mathit{jump}} \quad (\text{if } t_m \in \mathbb{R}),$$
$$((n, \bot, \bot), (\boldsymbol{v}, T_C)) \xrightarrow{\jmath_M} ((n, \bot, \bot), (\boldsymbol{v}, T_0)) \in \overline{\mathit{jump}}_1,$$

happen, where $T_0 = 0$ for the first step, and $T_0 = \epsilon$ for the other steps, and when $\tau_{src} < T_C$, the machine jumps (that generates any pair $(t_M, t_m) \in \mathbb{R}$ for the output) happen, where $u < T_C$:

$$\begin{aligned} & ((n, \bot, \bot), (T, u)) \xrightarrow{\mathfrak{I}_M} ((n+1, t_M, t_m), (0, u)) \in \mathit{jump} \\ \iff\ & t_M, t_m \in \mathbb{R} \ \wedge\ T = c_{\texttt{Main}}(n+1) - c_{\texttt{Main}}(n) \end{aligned}$$

The time-invariant constraint $(\forall t)\ x_{o_1}(t) = x_2(t)\ \wedge\ x_{o_2}(t) = x_1(t)$ for $\mathcal{E}$ can be encoded in $\bar{H}_{M_1 \restriction E_{M_1}}$ and $\bar{H}_{M_2 \restriction E_{M_2}}$ by renaming the variable x_{o_1} to x_2, and renaming the variable x_{o_2} to x_1.[23] Then, the ensemble hybrid automaton $H_{\mathcal{E}}$ is given by the composition $\bar{H}_{M_1 \restriction E_{M_1}} \parallel \bar{H}_{M_2 \restriction E_{M_2}} \parallel \bar{H}_{\texttt{Main}}$.

To illustrate, when $T_1 = 3, T_2 = 2$, and $T_{\mathcal{E}} = T_{\texttt{Main}} = 6$, given a maximal clock skew $0 \leq \epsilon < 2$, an initial state of $H_{\mathcal{E}}$ can be $((m_{\texttt{on}}, 0, \bot, \bot), (v_1, v_2, 0, 6+\epsilon)) \parallel ((m_{\texttt{on}}, 0, \bot, \bot), (v_2, v_1, 0, 6+\epsilon)) \parallel ((0, t_M, t_m), (0, 6+\epsilon))$. Since $\tau_{src} = T_{\mathcal{E}} + \epsilon$ for every automaton, communication jumps can happen until all the input buffers are full and all the output fullers are empty. For example, there exist:

$$\begin{aligned} ((m_{\texttt{on}}, 0, \bot, \bot), (v_1, v_2, 0, 6+\epsilon)) &\xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\max})}(t_M)} ((m_{\texttt{on}}, 0, t_M, \bot), (v_1, v_2, 0, 6+\epsilon)) \in \overline{\mathit{jump}}_1 \\ ((m_{\texttt{on}}, 0, \bot, \bot), (v_2, v_1, 0, 6+\epsilon)) &\xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\max})}(t_M)} ((m_{\texttt{on}}, 0, t_M, \bot), (v_1, v_2, 0, 6+\epsilon)) \in \overline{\mathit{jump}}_2 \\ ((0, t_M, t_m), (0, 6+\epsilon)) &\xrightarrow{\mathit{out}_{(\texttt{Main}, t_{\max})}(t_M)} ((0, \bot, t_m), (0, 6+\epsilon)) \in \overline{\mathit{jump}}, \end{aligned}$$

and therefore we have the next composed state $((m_{\texttt{on}}, 0, t_M, \bot), (v_1, v_2, 0, 6+\epsilon)) \parallel ((m_{\texttt{on}}, 0, t_M, \bot), (v_2, v_1, 0, 6+\epsilon)) \parallel ((0, \bot, t_m), (0, 6+\epsilon))$.

[23] In general, we can add all the variables in time-invariant constraints into the variable set of each hybrid automata, and make the constraints as their invariants.