Worst-Case Analysis of Digital Control Loops with Uncertain Input/Output Timing

Maximilian Gaukler and Peter Ulbrich

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
max.gaukler@fau.de, peter.ulbrich@fau.de

Abstract

Benchmark Proposal: The implementation of digital control systems in complex multicore or distributed real-time systems results in non-deterministic input/output timing. Such timing deviations typically lead to degraded performance or even instability, which in turn may jeopardize safety goals. We present the problem of proving worst-case guarantees for given input/output timing bounds as a benchmark for the verification of hybrid dynamical systems.

1 Introduction

Digital control implements feedback control on a computing system. Its defining characteristic is that reading sensor inputs and actuating the resulting control signal are performed at discrete points in time. Therefore, a common design assumption is the simultaneous and strictly periodic execution of these two steps. As automatic control is particularly sensitive to timing variations, a real-time computing system is used in safety-critical applications to assure temporal properties and thus a predictable quality of control.

However, while deterministic timing simplifies the design of the control system, its implementation becomes increasingly difficult from a real-time systems' perspective. Recent developments, such as distributed real-time systems, multicore platforms, and smart sensors, come at the cost of increasing complexity and degrading predictability. Consequently, the synchronous design approach to digital control becomes excessively expensive. A common solution to this issue is to resort to input/output (IO) windows, that is timing bounds instead of deterministic time instants. In turn, we have to verify that the resulting timing variations do no jeopardize the aspired safety and stability of the feedback control even in the worst case.

While stochastic simulations are a pragmatic and straightforward approach, they are generally incapable of proving worst-case properties, due to the infeasible number of possible execution flows and timings. Instead, we opt for a sound approximation of worst-case behavior by verification methods. To apply these methods, we model control loops with uncertain IO timing as hybrid dynamical systems. In a single model, this formalism allows combining the discrete-time aspects of a digital real-time control system with the continuous-time physics of a plant. Our preliminary experiments with existing verification tools indicate that solving this problem is feasible in some cases yet challenging in general.

In this paper, we present worst-case verification of digital control loops with uncertain IO timing as a benchmark problem for the verification of hybrid automata. Therefore, we detail the system model in section 2 and translate it to a hybrid system in section 3. Finally, formal goals and multiple examples are introduced in section 4.

2 System Model

For a formal description of the problem, we adapt the timing model of [10], which was used for stochastic average-case analysis, to the setting of worst-case stability verification. The main differences are a generalisation to nonlinear systems and the removal of reference tracking.

Plant The physical system to be controlled is described as a time-invariant plant, for which the following nonlinear equations and their linear special case, denoted by $\overset{\text{lin}}{=}$, are given.

$$\begin{aligned}\dot{x}_\text{p}(t) &= f_\text{p}(x_\text{p}(t), u(t), d(t)) \overset{\text{lin}}{=} A_\text{p}x_\text{p}(t) + B_\text{p}u(t) + G_\text{p}d(t), \quad t > 0,\\ y(t) &= g_\text{p}(x_\text{p}(t), d(t)) \overset{\text{lin}}{=} C_\text{p}x_\text{p}(t) + H_\text{p}d(t).\end{aligned} \tag{1}$$

The plant has input $u(t) = \begin{bmatrix}u_1(t) & u_2(t) & \dots & u_m(t)\end{bmatrix}^T \in \mathbb{R}^m$, state $x_\text{p}(t) \in \mathbb{R}^{n_\text{p}}$ and output $y(t) = \begin{bmatrix}y_1(t) & \dots & y_p(t)\end{bmatrix}^T \in \mathbb{R}^p$. A_p, B_p et cetera are matrices of appropriate dimension. The initial state $x_\text{p}(0) \in X_{\text{p},0} \subset \mathbb{R}^{n_\text{p}}$ is unknown but bounded. Disturbance and measurement noise are modeled by a bounded, nondeterministic input $d(t) \in D \subset \mathbb{R}^{n_\text{dist}}$. If they are not present, this will later be denoted by $n_\text{dist} = 0$, which means that the dependency on $d(t)$ and the terms $G_\text{p}d(t)$ and $H_\text{p}d(t)$ are omitted.

Controller The plant is controlled by the discrete-time controller

$$\begin{aligned}x_\text{d}[k+1] &= f_\text{d}(x_\text{d}[k], y[k]) \overset{\text{lin}}{=} A_\text{d}x_\text{d}[k] + B_\text{d}y[k], \quad x_\text{d} \in \mathbb{R}^{n_\text{d}},\\ u[k] &= g_\text{d}(x_\text{d}[k]) \overset{\text{lin}}{=} C_\text{d}x_\text{d}[k], \quad x_\text{d}[0] = 0, \quad k \in \mathbb{N}_0\end{aligned} \tag{2}$$

with nominal period T. This formulation includes standard linear controllers, such as discretized PID or observer-based state feedback. The given form has no feedthrough, which means that the control signal $u[k]$ only requires the previous measurement $y[k-1]$. This restriction is reasonable for control systems in which timing is relevant, as it permits a computation time of slightly below one control period, while with feedthrough any computation time would inevitably delay the output. For simplicity and due to the goal of stability analysis, we only consider a constant set point of $y = 0$.

IO timing model Nominally, the whole measurement vector $y[k]$ is sampled at $t = kT$. From this, the control signal $u[k+1]$ is computed and then emitted at $t = (k+1)T$. The real timing differs from this: The j-th control output component is delayed by $\Delta t_{\text{u},j}[k]$ (where negative values represent a too early output), which can formally be stated as

$$u_j(t) = \begin{cases} u_j[k], & kT + \Delta t_{\text{u},j}[k] < t < (k+1)T + \Delta t_{\text{u},j}[k+1] \quad \wedge \quad k \geq 1 \\ 0, & \text{else.}\end{cases} \tag{3}$$

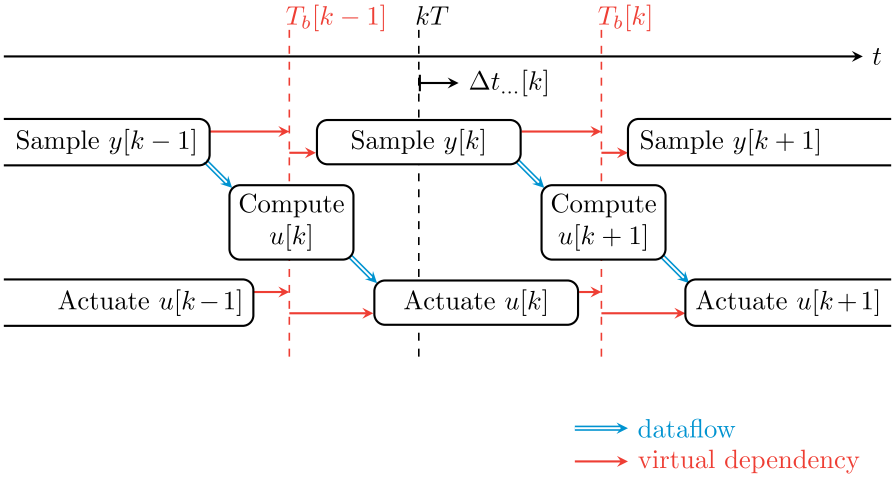

Figure 1: Timing model for IO and computation with dataflow dependencies, the timing barrier $T_\mathrm{b}[k]$ and its corresponding virtual dependencies.

Respectively, the sample $y_j[k]$ of the j-th sensor is acquired with a time offset $\Delta t_{\mathrm{y},j}[k]$.

$$y_j[k] = \begin{cases} y_j(kT + \Delta t_{\mathrm{y},j}[k]), & k \geq 1 \\ 0, & \text{else.} \end{cases} \tag{4}$$

The "0" case in eqs. (3) and (4) is only relevant for start-up and will be discussed later.

The timing of input, computation and output and its dataflow dependencies (double arrows) are shown in fig. 1. As a "black-box" model, the timing of each input and output component is unknown but bounded:

$$\underline{\Delta t}_{\mathrm{u},j} \leq \Delta t_{\mathrm{u},j}[k] \leq \overline{\Delta t}_{\mathrm{u},j}, \quad \underline{\Delta t}_{\mathrm{y},j} \leq \Delta t_{\mathrm{y},j}[k] \leq \overline{\Delta t}_{\mathrm{y},j} \qquad \forall j, k \geq 1 \tag{5}$$

Additional constraints, e. g., that a group of inputs is always sampled at the same time, may be considered to make the model less pessimistic, but are not discussed here for the sake of clarity.

To avoid the need for extra buffers in the realization and corresponding buffer states in the resulting model, neighboring cycles k and $k+1$ are separated by a timing barrier $T_\mathrm{b}[k]$, which no event may cross:

$$\left.\begin{matrix} kT + \Delta t_{\mathrm{y},j}[k] \\ kT + \Delta t_{\mathrm{u},j}[k] \end{matrix}\right\} < T_\mathrm{b}[k] < \left\{\begin{matrix} (k+1)T + \Delta t_{\mathrm{y},j}[k+1] \\ (k+1)T + \Delta t_{\mathrm{u},j}[k+1] \end{matrix}\right. \qquad \forall j, k \geq 1 \tag{6}$$

This barrier coincides with the controller computation and can equivalently be described by a task with zero execution time and four virtual dependencies as shown in fig. 1. To simplify the subsequent model, we only consider $T_\mathrm{b} = (k + \frac{1}{2})T$ in the following, which means that each control period k is centered at its nominal time kT and all delays are limited to $|\Delta t_{\dots}| < \frac{T}{2}$.

Startup behavior Because time doesn't start at $t = -T/2$, but at $t = 0$, the first half of the cycle $k = 0$ is missing, and $y_j[0]$ would not be sampled if $\Delta t_{\mathrm{y},j}[0] < 0$. To avoid this discontinuity, sampling $y[0]$ and actuating $u[0]$ are skipped, which is reflected in the "0" case

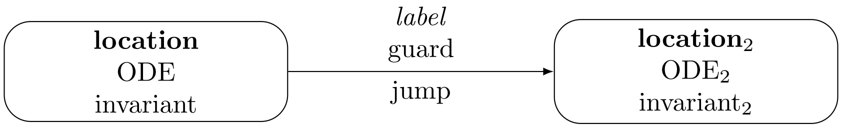

Figure 2: Legend for hybrid automata used in this publication

of eqs. (3) and (4). Because the startup behavior only affects a finite time interval at the start, changing it is roughly equivalent to a change of the initial set. For the linear case, it is therefore not relevant for infinite-time, e. g. asymptotic stability.

3 Hybrid Automata

The system model presented in the previous section can be equivalently formulated as the interconnection of multiple hybrid automata, which is a precise and machine-readable formal description suitable for automatic verification. A simplified summary of the semantics of hybrid automata will be given in the following. We refer to [8] for an extended introduction and to [11] for a strict formalization.

Semantics A hybrid automaton as shown in fig. 2 combines a discrete-event automaton with a classical continuous-time, continuous-valued dynamical system. In each location (discrete-valued state) of the discrete-event automaton, the state variables of the continuous system evolve according to an ordinary differential equation (ODE) and are restricted to a set given by an invariant condition. A transition to another location *may* be taken while the associated guard condition is true (*may-semantics*) and *must* be taken before the invariant condition of the current location is violated. At the transition, the continuous state changes instantaneously as given by the jump transition, e. g., $x' = 2x$ states that x jumps to twice its previous value. For synchronization of multiple automata, labels are used: If multiple transitions have the same nonempty label, either all are executed at once or none of them.

Nondeterminism Because the timing is bounded but otherwise unknown, the system is nondeterministic: Multiple trajectories are possible for a given disturbance and initial state. The nondeterminism of *may-semantics* mentioned before is used to represent the nondeterminism of IO timing. This would be significantly more difficult if *must-semantics (urgent semantics)* were used instead, which defines that transitions must be taken as soon as possible.

3.1 Components

For a modular and comprehensible description, the system is modeled as the composition of components that are defined in the following: Controller and clock, the physical plant and multiple sample-and-hold (SH) elements.

Controller and Clock We define a periodic sawtooth clock signal

$$\tau = ((t + T/2) \mod T) - T/2, \tag{7}$$

which is the signed distance $t - kT$ to the nearest nominal sampling point kT. This signal is generated by the automaton shown in fig. 3, which also includes the controller: τ continuously

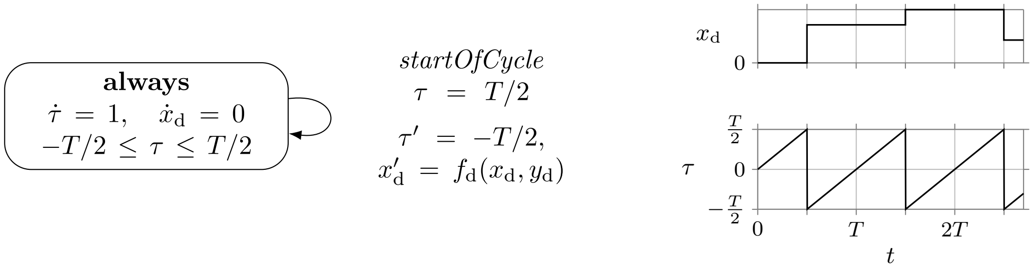

Figure 3: Hybrid automaton (left) and example simulation (right) of controller and clock

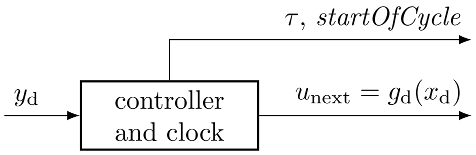

Figure 4: Block diagram of controller and clock

increases ($\dot{\tau} = 1$) until $\tau = T/2$, which is the timing boundary $t = (k + 1/2)T$ betweeen two control cycles, is reached. Then, τ is reset from $T/2$ to $-T/2$ and the new controller state is computed from the recent measurement. This event is given a synchronization label *startOfCycle*, which will later be used to force the reset of all sample-and-hold automata.

A block diagram representation is shown in fig. 4. The input is the sampled measurement y_d. The outputs are the clock τ, the label *startOfCycle*, which corresponds to the falling edge of τ, and the “next” (most recently computed) control signal $u_\mathrm{next} = g_\mathrm{d}(x_\mathrm{d})$.

Generic sample-and-hold (SH) The transitions between discrete-time controller and continuous-time plant can be modeled by SH elements as shown in fig. 5. The output b is updated to the value of the input a once per period, within $\underline{\Delta t} \le \tau \le \overline{\Delta t}$, and held constant inbetween ($\dot{b} = 0$). In accordance with the timing model, the parameters $\overline{\Delta t}$ and $\underline{\Delta t}$ are restricted to $-T/2 < \overline{\Delta t} \le \underline{\Delta t} < T/2$.

This description is implemented by the automaton in fig. 6, which uses may-semantics to model that sampling *may* happen while $\underline{\Delta t} \le \tau \le \overline{\Delta t}$ and *must* happen before $\tau > \overline{\Delta t}$. The *startOfCycle* synchronization label, which occurs at the transition from $\tau = T/2$ to $\tau = -T/2$, is used to reset the automaton at the beginning of each cycle.

Plant The plant is described the equations given in section 2:

$$\dot{x}_\mathrm{p} = f(x_\mathrm{p}, u, d), \quad y = g_\mathrm{p}(x_\mathrm{p}, d), \quad d \in D. \tag{8}$$

3.2 Interconnection

The closed loop is modeled by connecting controller and plant with one sample-and-hold block per input and output component as shown in fig. 7. All timing-dependent components share the common clock τ and the corresponding *startOfCycle* synchronization label.

In this block diagram, controller and plant are interconnected with $u_\mathrm{next} = g_\mathrm{d}(x_\mathrm{d})$ and $y = g_\mathrm{p}(x_\mathrm{p}, d)$. As such interconnection variables which are not needed as state variables may

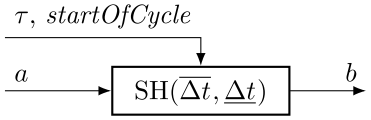

Figure 5: Block diagram of sample-and-hold element

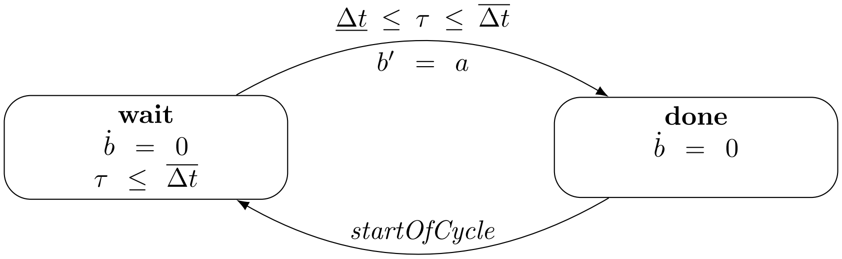
$\underline{\Delta t} \leq \tau \leq \overline{\Delta t}$
$b' = a$
wait
$\dot{b} = 0$
$\tau \leq \overline{\Delta t}$
done
$\dot{b} = 0$
startOfCycle

Figure 6: Hybrid automaton of sample-and-hold element

introduce additional computational effort in analysis, a different but equivalent implementation was used for the experiments described later: The states x_d and x_p were used as interconnection signal and the computation of $g_\mathrm{p}(...)$ and $g_\mathrm{d}(...)$ was moved into the respective SH components.

3.3 Initialization and Startup

The controller is initialized at $x_\mathrm{d} = 0, \tau = 0$, and all SH automata start in location "done" with state $b = 0$. The plant state is bounded by $x_\mathrm{p}(0) \in X_{\mathrm{p},0}$. The SH automata are initialized at location "done" to match to the specified behavior of skipping the cycle $k = 0$.

Alternative startup behavior A different startup behavior could be modeled by starting at $\tau = -T/2$ and "wait", which would be equal to starting at $t = T/2$ with the first regular cycle $k = 1$. It is not generally possible to initialize $\tau = 0$ and still use "wait" as initial location, because the invariant $\tau \leq \overline{\Delta t}$ would be violated at start if $\overline{\Delta t} < 0$.

4 Benchmark Setup

For safety-critical real-time control, it is desirable to automatically validate the system's correctness, for example, that a quadcopter does not crash. In the following, we translate this vision into a formal problem with goals that can be checked using tools for the verification of hybrid systems. To facilitate evaluation and comparison, we provide a metric and parameters for example systems as well as first experimental results.

Given A digital control loop with uncertain but bounded input/output timing and disturbance as modeled in section 2, equivalent to the hybrid system described in section 3.

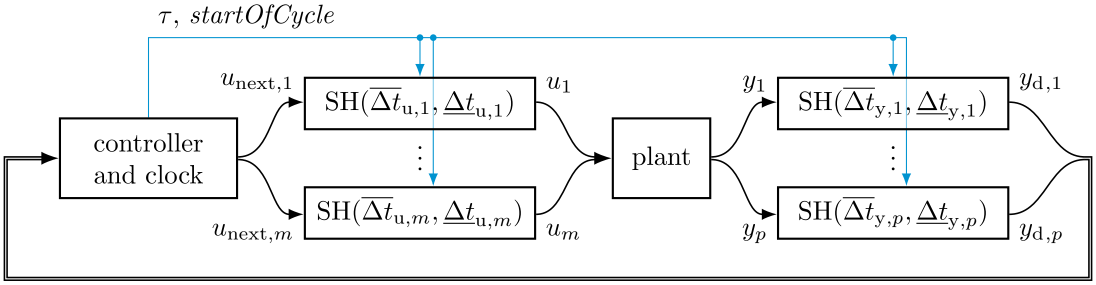
τ, *startOfCycle*
$u_{\text{next},1}$
$\mathrm{SH}(\overline{\Delta t}_{\mathrm{u},1}, \underline{\Delta t}_{\mathrm{u},1})$
u_1
y_1
$\mathrm{SH}(\overline{\Delta t}_{\mathrm{y},1}, \underline{\Delta t}_{\mathrm{y},1})$
$y_{\mathrm{d},1}$
controller and clock
plant
$u_{\text{next},m}$
$\mathrm{SH}(\overline{\Delta t}_{\mathrm{u},m}, \underline{\Delta t}_{\mathrm{u},m})$
u_m
y_p
$\mathrm{SH}(\overline{\Delta t}_{\mathrm{y},p}, \underline{\Delta t}_{\mathrm{y},p})$
$y_{\mathrm{d},p}$

Figure 7: Block diagram of closed control loop

Goal 1: Stability proof Prove practical stability of the control loop, which we define as that for bounded initial state, the infinite-time reachable set

$$S := \left\{ \left[x_{\mathrm{p}}^T(t)\;\; x_{\mathrm{d}}^T(t)\;\; u^T(t)\;\; y^T(t)\right]^T \middle|\; t \geq 0,\ \text{dynamics of section 3, with uncertainties } x_{\mathrm{p}}(0) \in X_{\mathrm{p},0},\ \Delta t_{\dots}[k] \in \left[\underline{\Delta t}_{\dots}; \overline{\Delta t}_{\dots}\right], d(\xi) \in D\, \forall \xi \right\} \qquad (9)$$

is bounded. For a quadcopter, this means that there is a proven, possibly large, bound on the worst-case position error. For any linear controller and plant, this implies marginal or exponential stability, if the internal timing τ is not considered a state.

Practical stability can be shown with set-valued reachability analysis by finding a *bounded* upper approximation $\bar{S} \supseteq S$ of the reachable set.

Goal 1b: Exponential decay Prove global uniform exponential stability of the controlled plant: For zero disturbance ($n_{\text{dist}} = 0$ or $D = \{0\}$), find constants $C, \lambda > 0$ such that

$$\|x_{\mathrm{p}}(t)\|_2 \leq C \mathrm{e}^{-\lambda t} \|x_{\mathrm{p}}(0)\|_2 \quad \forall t \geq 0, \quad \forall x_{\mathrm{p}}(0). \qquad (10)$$

While practical stability only ensures that the state is bounded, exponential stability is desirable because it guarantees a certain rate of decay for the initial error.

Goal 2: Tight bounds If the system is stable, compute useful bounds on the states $x_{\mathrm{p}}, x_{\mathrm{d}}, u, y$: For a quadcopter, the analysis must prove that worst-case position error is less than a few centimeters, not kilometers. In general, this guaranteed and therefore outer bound $\bar{S} \supseteq S$ obtained from analysis should be not much larger than an inner approximation $\underline{S} \subseteq S$ from a large number of random simulations. As a metric for usefulness, we introduce the bloating factor K, the worst ratio of the component-wise bounds of analysis and simulation:

$$K := \max_j \frac{\max\limits_{x \in \bar{S}} \left|e_j^T x\right|}{\max\limits_{x \in \underline{S}} \left|e_j^T x\right|}, \quad \text{where } e_j \text{ denotes the } j\text{-th unity vector.} \qquad (11)$$

An equivalent definition is illustrated in fig. 8:

$$\boxdot(\bar{S}) \overset{!}{\subseteq} K \boxdot(\underline{S}) \quad \text{with minimal } K, \qquad (12)$$

where $\boxdot(\cdot)$ denotes the enclosing symmetric multidimensional interval (symmetric box).

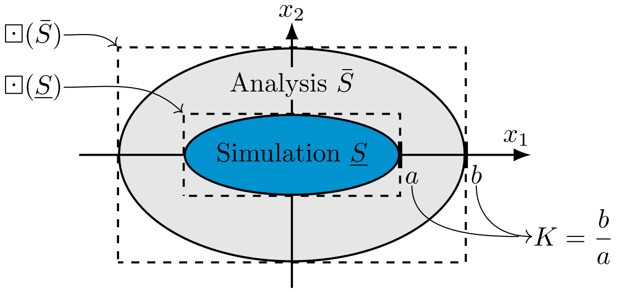

Figure 8: Illustration of the metric K, which compares the interval bounds of simulation and pessimistic analysis.

An ideal verification tool can achieve $K = 1$ in short computation time. $K \gg 1$ corresponds to a large undesirable gray area between analysis and simulation, in which the existence of trajectories can neither be confirmed nor denied.

Goal 2b: Extreme examples Efficiently finding extreme simulation traces to show minimal K is a problem on its own. Ideally, a verification tool should provide concrete examples of the worst case to show that its result is not unnecessarily pessimistic.

4.1 Approximate Continuous-Time Variant

For comparison, we provide a purely continuous variant of the problem: Assuming perfect timing and a negligibly small period T, the discrete-time difference quotient of the controller state x_d is approximated by a continuous-time differential equation. With $t = kT$,

$$\dot{x}_\mathrm{d}(t) \approx \frac{x_\mathrm{d}[k+1] - x_\mathrm{d}[k]}{T} = \frac{f_\mathrm{d}(x_\mathrm{d}[k], y[k]) - x_\mathrm{d}[k]}{T} \overset{\mathrm{lin}}{=} \frac{A_\mathrm{d} - I}{T} x_\mathrm{d}(t) + \frac{B_\mathrm{d}}{T} y(t). \tag{13}$$

The new closed loop does not require states for measurement and actuation and is given by

$$\begin{bmatrix} \dot{x}_\mathrm{p}(t) \\ \dot{x}_\mathrm{d}(t) \end{bmatrix} \overset{\mathrm{lin}}{=} \begin{bmatrix} A_\mathrm{p} & B_\mathrm{p} C_\mathrm{d} \\ T^{-1} B_\mathrm{d} C_\mathrm{p} & T^{-1}(A_\mathrm{d} - I) \end{bmatrix} \begin{bmatrix} \dot{x}_\mathrm{p}(t) \\ \dot{x}_\mathrm{d}(t) \end{bmatrix} + \begin{bmatrix} G_\mathrm{p} \\ B_\mathrm{d} H_\mathrm{p} \end{bmatrix} d(t). \tag{14}$$

We remark that the stability of this system is neither sufficient nor necessary for the stability of the original, discrete-time system. However, modeling the difference between (14) and the original system as bounded disturbance could yield a continuous abstraction (continuization) of the original controller, similar to [5].

Examples and Experiments Appendix B provides data for several examples ranging from a minimal, one-dimensional system to the simplified model of three-axis angular rate control of a quadcopter. Experiments with a number of tools based on set-valued reachability analysis were carried out and showed that stability could only be verified for the most trivial examples, whereas all other examples led to errors or did not finish within a generous timeout. A detailed description of the experiments can be found in appendix C. Model files and open-source code for reproducing the experiments and generating new examples can be found on-line (see appendix A).

5 Summary and Outlook

Uncertain input/output timing adversely affects the performance of digital control loops but is virtually unavoidable in complex real-time systems. To still prove safety we are required to prove worst-case behavior, such as the maximum position error of a quadcopter, in the presence of timing uncertainty. To address this challenge, we present a series of benchmark problems for the verification of hybrid automata, which are a formalism that captures both the discrete-time and continuous-time aspects of real-time control systems. However, first experiments with set-based reachability tools suggest that verification is only possible for simple examples.

Set-based reachability analysis typically performs overapproximations at discrete transitions to keep computational effort acceptable. This suggests that it is best suited for systems with few discrete transitions and stable continuous dynamics; the opposite of this is the case for digital control loops. Our experiment substantiates this hypothesis, raising the question of whether we are faced with a fundamental problem or if a tailored verification approach or modified algorithms can solve the issues.

While numerous techniques for the verification of discrete-time controllers exist, to the best of our knowledge, none of them supports timing uncertainties as presented in our model, which addresses real-time systems with multiple sensors and actuators. Therefore, future work will entail an extension of existing techniques. One candidate is continuization [5], a method for approximating discrete-time controllers as continuous-time controllers with disturbance, which are more suitable for reachability analysis. Further candidates are LMI-based methods for linear control systems as discussed in [12].

References

[1] Matthias Althoff. An introduction to CORA 2015. In Goran Frehse and Matthias Althoff, editors, *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, volume 34 of *EPiC Series in Computing*, pages 120–151. EasyChair, 2015.

[2] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST. In *Proceedings of the 18th International Conference on Hybrid Systems Computation and Control - HSCC '15*. ACM Press, 2015.

[3] Stanley Bak and Marco Caccamo. Computing reachability for nonlinear systems with HyCreate. 16th International Conference on Hybrid Systems: Computation and Control Poster Session, 2013.

[4] Stanley Bak and Parasara Sridhar Duggirala. HyLAA: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, pages 173–178, New York, NY, USA, 2017. ACM.

[5] Stanley Bak and Taylor T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *2015 IEEE Real-Time Systems Symposium*. IEEE, 2015.

[6] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

[7] A. E. C. Da Cunha. Benchmark: Quadrotor attitude control. In Goran Frehse and Matthias Althoff, editors, *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, volume 34 of *EPiC Series in Computing*, pages 57–72. EasyChair, 2015.

[8] Goran Frehse. An introduction to hybrid automata, numerical simulation and reachability analysis. In *Formal Modeling and Verification of Cyber-Physical Systems*, pages 50–81. Springer Fachmedien Wiesbaden, 2015.

[9] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer Berlin Heidelberg, 2011.

[10] Maximilian Gaukler, Andreas Michalka, Peter Ulbrich, and Tobias Klaus. A new perspective on quality evaluation for control systems with stochastic timing. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week) - HSCC '18*. ACM Press, 2018.

[11] T.A. Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292. IEEE Comput. Soc. Press, July 1996.

[12] Laurentiu Hetel, Christophe Fiter, Hassan Omran, Alexandre Seuret, Emilia Fridman, Jean-Pierre Richard, and Silviu Iulian Niculescu. Recent developments on the stability of systems with aperiodic sampling: An overview. *Automatica*, 76:309–335, February 2017.

A Source Code and Data Files

The source code and output data used in this publication is available online at `https://doi.org/10.5281/zenodo.2600139`. Updated versions will be provided at `https://github.com/qronos-project/arch19-benchmark-iotiming`.

B Example Data

In the following, example systems are presented. For reference, the systems are denoted by a letter, followed by a number indicating the variant. To limit implementation complexity and the number of variants, we restrict the examples to linear systems and only consider variants without disturbance in the subsequent experiments.

Note: SpaceEx model files for this section can be found in `code/template/output/`, e. g., `code/template/output/solved_with_spaceex/stable/A1_1.spaceex.xml` for example A1.

B.1 One-dimensional Example (A1)

For an example of minimal dimensions $n = n_\mathrm{d} = m = p = 1$, a linear, weakly unstable plant $A_\mathrm{p} = 0.05, B_\mathrm{p} = 0.5, C_\mathrm{p} = 1$, without disturbance ($n_\mathrm{dist} = 0$) is controlled by the discrete-time controller $A_\mathrm{d} = -0.01, B_\mathrm{d} = -0.4, C_\mathrm{d} = 1$. The timing is uncertain ($\underline{\Delta t}_y = -0.1, \overline{\Delta t}_y = 0.002, \underline{\Delta t}_u = -0.001, \overline{\Delta t}_u = 0.002$) and the initial state is bounded within the interval $X_{\mathrm{p},0} = [-1; 1]$. It should be noted that even for this one-dimensional plant and controller, the resulting hybrid system has 4 continuous state variables $(x_\mathrm{p}, u, y_\mathrm{d}, x_\mathrm{d})$.

The parameters of this system were heuristically chosen such that SpaceEx can verify stability via an invariant set within seconds with optimized settings and within minutes with almost arbitrary settings. Variants with increased and decreased difficulty are also provided:

A2 Negligible jitter: $\overline{\Delta} t_{u,y} = -\underline{\Delta} t_{u,y} = 0.0001$

A3 Increased jitter: $\overline{\Delta} t_u = -0.4, \underline{\Delta} t_u = 0.1, \overline{\Delta} t_u = -0.1, \underline{\Delta} t_u = 0.4$

A4 Extreme jitter: Variant A3 with $\overline{\Delta} t_u = 0.4$

A5 Increased dimension: Variant A3 is duplicated, which means that all dynamics matrices are diagonally repeated such as $A_{\mathrm{p},A5} = \mathrm{diag}(A_{\mathrm{p},A3}, A_{\mathrm{p},A3})$. This increases the dimension by factor 2, making verification more difficult, but does not change stability.

A6 Disturbance (not included in experiments): To add disturbance and measurement noise, the system may be changed to $n_{\text{dist}} = 2, G_{\text{p}} = \begin{bmatrix}1 & 0\end{bmatrix}, H_{\text{p}} = \begin{bmatrix}0 & 1\end{bmatrix}, D = [-1; 1]^2$.

B.2 Trivial Three-dimensional Example (B1)

For an example of higher order, a stable three-dimensional plant is combined with a controller that has negligible influence on the plant dynamics:

$$A_{\text{p}} = \begin{bmatrix} -1 & 0.002 & 0.003 \\ 0.004 & -5 & 0.006 \\ 0.007 & 0.008 & -9 \end{bmatrix}, \quad B_{\text{p}} = \begin{bmatrix} 0.001 & 0.0011 \\ 0.0012 & 0.0013 \\ 0.0014 & 0.0015 \end{bmatrix}, \quad C_{\text{p}} = \begin{bmatrix} 16 & 17 & 18 \end{bmatrix} \tag{15}$$

$$A_{\text{d}} = \begin{bmatrix} 0.019 & 0.02 \\ 0.021 & 0.022 \end{bmatrix}, \quad B_{\text{d}} = \begin{bmatrix} 0.023 \\ 0.024 \end{bmatrix}, \quad C_{\text{d}} = \begin{bmatrix} 0.025 & 0.026 \\ 0.027 & 0.028 \end{bmatrix} \tag{16}$$

$$T = 2, \quad \overline{\Delta t}_u = -\underline{\Delta t}_u = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}, \quad \overline{\Delta t}_y = -\underline{\Delta t}_y = 0.6, \quad X_{\text{p},0} = [-1; 1]^3, \quad n_{\text{dist}} = 0 \tag{17}$$

Variant B2: Disturbance (not included in experiments) To add input disturbance, the system may be changed to $n_{\text{dist}} = 1, G_{\text{p}} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T, H_{\text{p}} = \begin{bmatrix}0 & 0\end{bmatrix}^T, D = [-1; 1]$.

B.3 Quadcopter Examples (C, D)

The linearized dynamics of the angular rate control of a quadcopter, mostly based on [7], are the basis for two models. Angular rate control is an important subsystem of quadcopter stabilization, which ensures that the rate of rotation r_ϕ, r_θ, r_ψ around the x, y and z axis follows the setpoint given by a higher-level control system. For simplicity, the latter is not considered here and instead the setpoint is assumed to be zero. For a detailed system description and physical model, we refer to [7] and the references provided there.

Single-axis angular rate control (C) If only the rotation ϕ around the x-axis, is considered, the mechanical dynamics of the angular rate r_ϕ are

$$J_\phi \dot{r}_\phi(t) = T_\phi(t) \tag{18}$$

with rotational inertia J_ϕ. The input is the time-varying torque $T_\phi(t)$ controlled by the motors and the output is the angular rate $r_\phi(t)$ measured by a gyroscope.

The original, continuous-time controller

$$T_\phi(t) = -K_{p,\phi} r_\phi(t) - K_{I,\phi} \int_0^t r_\phi(\xi) \mathrm{d}\xi \tag{19}$$

given in [7] is discretized in a simplified way as

$$T_\phi[k] = -K_{P,\phi} r_\phi[k-1] - K_{I,\phi} \sum_0^{k-1} r_\phi[k]. \tag{20}$$

To match the timing model, $T_\phi[k]$ has no feedthrough, i. e., it does not depend on the current measurement $r_\phi[k]$.

Three-axis angular rate control (D) If all three axes of rotation are considered, the linearized dynamics are three independent integrators

$$J_\phi \dot{r}_\phi(t) = T_\phi(t), \tag{21}$$
$$J_\theta \dot{r}_\theta(t) = T_\theta(t), \tag{22}$$
$$J_\psi \dot{r}_\psi(t) = T_\psi(t). \tag{23}$$

The torques $T_{\phi,\theta,\psi}$ and the vertical thrust F_z are controlled by the forces $u_{1,2,3,4}$ of four propellers, which are defined as inputs to the system because rotor dynamics are neglected:

$$\begin{bmatrix} T_\phi(t) \\ T_\theta(t) \\ T_\psi(t) \\ F_z(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ \gamma & -\gamma & \gamma & -\gamma \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix}}_{u(t)}. \tag{24}$$

This equation depends on the distance l between center and rotor and the ratio γ of torque to rotor thrust.

Because altitude control is not considered here, we set $F_z = 0$ and obtain

$$u(t) = M^{-1} \begin{bmatrix} T_\phi(t) \\ T_\theta(t) \\ T_\psi(t) \\ 0 \end{bmatrix}, \tag{25}$$

where $T_{\phi,\theta,\psi}(t)$ is each computed by a PI controller in the structure of eq. (20), but with axis-dependent controller parameters.

It should be noted that for perfect timing, the system can be split into three decoupled subsystems, whereas if the components of $u(t)$ are not updated synchronously, the control torque of one axis slightly affects other axes.

While the presented model is highly simplified, our experiments in verifying it were not successful. As soon as it has been successfully verified, more realistic variants should be considered, e. g., by adding disturbance, nonlinearities, rotor dynamics and position control.

Parameters The original publication [7] states the following parameters: $J_\phi = 9.036{\cdot}10^{-6}$, $J_\theta = 9.127{\cdot}10^{-6}$, $J_\psi = 1.937{\cdot}10^{-5}$, $K_{P,\phi} = 2.557{\cdot}10^{-4}$, $K_{P,\theta} = 2.581{\cdot}10^{-4}$, $K_{P,\psi} = 5.478{\cdot}10^{-4}$, $K_{I,\phi} = 3.614{\cdot}10^{-3}$, $K_{I,\theta} = 3.651{\cdot}10^{-3}$, $K_{I,\psi} = 7.747{\cdot}10^{-3}$.

We assume $l = 0.1$ and $\gamma = 0.01$, which are not given in [7] and therefore chosen within the order of magnitude observed in other quadcopter models. The controller period is chosen as $T = 0.01$, which is a compromise between a response similar to the original continuous-time controller and low processor utilization. The initial response of the original and the discretized controller are compared in fig. 9.

The dynamics matrices for example C are

$$A_\mathrm{p} = 0, \quad B_\mathrm{p} = 1.107{\cdot}10^{5}, \quad C_\mathrm{p} = 1, \tag{26}$$
$$A_\mathrm{d} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B_\mathrm{d} = \begin{bmatrix} 1{\cdot}10^{-2} \\ 1 \end{bmatrix}, \quad C_\mathrm{d} = \begin{bmatrix} -3.614{\cdot}10^{-3} & -2.556{\cdot}10^{-4} \end{bmatrix}, \tag{27}$$

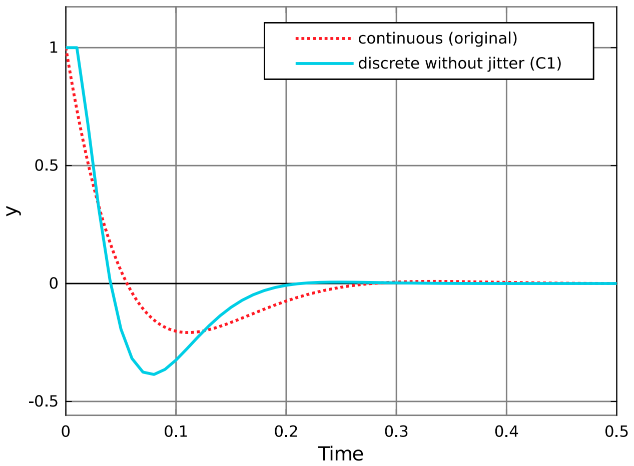

Figure 9: Initial response $r_\phi(t)$ of original and discretized controller for example C for an initial state $r_\phi(0) = 1$ and perfect timing.
(Simulink simulation: `code/template/example_C_simulation_of_nominal_case.slx`)

and for example D are

$$A_\mathrm{p} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_\mathrm{p} = \begin{bmatrix} 0 & -1.107{\cdot}10^{4} & 0 & 1.107{\cdot}10^{4} \\ -1.096{\cdot}10^{4} & 0 & 1.096{\cdot}10^{4} & 0 \\ 5.163{\cdot}10^{2} & -5.163{\cdot}10^{2} & 5.163{\cdot}10^{2} & -5.163{\cdot}10^{2} \end{bmatrix}, \tag{28}$$

$$C_\mathrm{p} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad A_\mathrm{d} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_\mathrm{d} = \begin{bmatrix} 1{\cdot}10^{-2} & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1{\cdot}10^{-2} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1{\cdot}10^{-2} \\ 0 & 0 & 1 \end{bmatrix}, \tag{29}$$

$$C_\mathrm{d} = \begin{bmatrix} 0 & 0 & 1.825{\cdot}10^{-2} & 1.291{\cdot}10^{-3} & -1.937{\cdot}10^{-1} & -1.370{\cdot}10^{-2} \\ 1.807{\cdot}10^{-2} & 1.279{\cdot}10^{-3} & 0 & 0 & 1.937{\cdot}10^{-1} & 1.370{\cdot}10^{-2} \\ 0 & 0 & -1.825{\cdot}10^{-2} & -1.291{\cdot}10^{-3} & -1.937{\cdot}10^{-1} & -1.370{\cdot}10^{-2} \\ -1.807{\cdot}10^{-2} & -1.279{\cdot}10^{-3} & -2.533{\cdot}10^{-18} & -1.791{\cdot}10^{-19} & 1.937{\cdot}10^{-1} & 1.370{\cdot}10^{-2} \end{bmatrix}. \tag{30}$$

We assume no disturbance ($n_\mathrm{dist} = 0$) and an initial state within $X_{\mathrm{p},0} = [-1; 1]^{n_\mathrm{p}}$. Two timing variants are provided:

C1, D1 Perfect timing: $\overline{\Delta} t_{u,y} = \underline{\Delta} t_{u,y} = 0$

C2, D2 The timing may vary by ±0.01T: $\overline{\Delta} t_{u,y} = -\underline{\Delta} t_{u,y} = 0.01T \begin{bmatrix} 1 & 1 & \dots \end{bmatrix}^T$.

B.4 Timer Example (E)

For debugging and visualization, the following example can be used. The plant is a timer with output $y(t) = x_{\mathrm{p},1}(t) = t$, regardless of the input. Therefore, it is obviously unstable. The controller is $u[k+1] = y[k]$. Because u has no effect on y, this results in $u[k+1] = kT + \Delta t_y[k]$. This means that $x_{\mathrm{p},1}$ is the time axis and the value of both u and $y[k]$ is the time t at which the measurement was sampled. The parameters are given in the following:

$$A_\mathrm{p} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B_\mathrm{p} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad C_\mathrm{p} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad X_{\mathrm{p},0} = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}, \quad n_\mathrm{dist} = 0 \tag{31}$$

$$A_\mathrm{d} = 0, \quad B_\mathrm{d} = 1, \quad C_\mathrm{d} = 1 \tag{32}$$

$$T = 1, \quad \underline{\Delta t}_u = -0.4, \quad \overline{\Delta t}_u = 0.1, \quad \underline{\Delta t}_y = -0.2, \quad \overline{\Delta t}_y = 0.3 \tag{33}$$

C Experiments

First, a selection of tools is evaluated in appendix C.1. All examples presented in this publication are then evaluated with the selected candidates, SpaceEx and Pysim, in appendix C.2.

Computation times were measured on a 3.6 GHz Intel Core i7-4790 CPU, running on Ubuntu 18.04 with a memory limit of 14 GB. For code, model files, tool settings and output can be found in appendix A.

C.1 Selection of Tools

We restricted our experiments to SpaceEx [9] and a small number of open-source set-valued reachability tools for which a conversion from SpaceEx exists. The SpaceEx tool was used as a starting point because it directly supports the modeling scheme used in section 3. The one-dimensional example A1 from appendix B.1 was created in the format of SpaceEx. For use with other tools, it was converted using Hyst [2]. As most other tools don't support the interconnection of multiple automata, this conversion also includes computing the synchronous product to yield a single equivalent automaton.

Note: Code and data for this section can be found in `code/dummy/`.

SpaceEx With manually optimized settings, which can be found in `code/dummy/a1.cfg`, SpaceEx 0.9.8f successfully finds the fixpoint for the reachable set shown in fig. 10 after few seconds. As mentioned earlier, the parameters of example A1 were specifically chosen such that verification with SpaceEx is possible, so this result is by design. Further experiments with SpaceEx can be found in the next section.

Flowstar By design, the Flowstar [6] tool only supports bounded-time reachability computation. As a workaround, convergence of the desired infinite-time reachable set was approximated by comparing the reachable set for finite time horizons from 5 to 100. The time step setting was varied from 10^{-1} to 10^{-4}, where the latter already requires one hour of computation time for a time horizon of 10.

We were unable to obtain useful results with Flowstar 2.0.0[1], even for negligible timing uncertainties as in example A2: For a time horizon of 10, the resulting set was already five times bigger than the infinite-time result of SpaceEx, and for larger time horizons Flowstar did not finish within a run-time limit of two hours. However, it cannot be completely ruled out that this is due to an unfortunate choice of parameters.

[1] A newer version 2.1.0 is available, but not yet supported by Hyst.

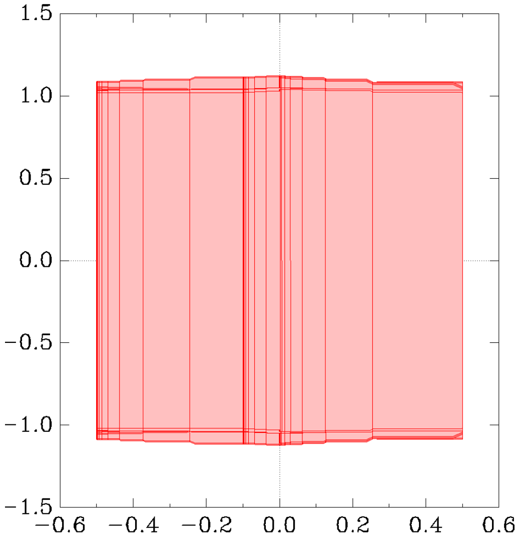

Figure 10: Reachable set (y over τ) computed by SpaceEx for example A1

HyCreate According to its website[2], the HyCreate [3] tool has been discontinued and is not recommended for more than three continuous states for longer time horizons. Nevertheless, a short experiment was carried out. After correcting syntax errors in Hyst's output, HyCreate 2.81 returned a reachable set that only covers $\tau > 0$, which means that the periodic reset transition $\tau' = -T/2$ is never taken. A similar result was shown by HyCreate's integrated simulation engine. After weakening the guard condition $\tau = T/2$ to $\tau \geq 0.9\,T/2$, the region $-T/2 < \tau < 0$ is reached by simulations, but still not contained in the computed reachable set (see fig. 11), which suggests an error in HyCreate or Hyst. The corresponding files can be found in `code/dummy/hycreate2/`.

Hylaa As of January 2019, the stable version of Hylaa [4] (55a72f8, "master" branch) does not yet support resets, which means the state must not change at a transition. Therefore Hylaa is not suitable for our model. While the current development version ("hybrid" branch) seems to support transitions, it could not be used because it is not yet supported by Hyst. However, it is a promising candidate for future experiments.

CORA Due to a misunderstanding, we assumed that CORA [1] does not support the may-semantics used by our model and therefore excluded it from our experiments. We will, however, include it in future experiments provided online.

Pysim The Pysim simulator supplied with Hyst uses must-semantics, as can be seen from its source code. However, as Pysim only simulates, this only means that it will constantly use the earliest possible timing, which is technically correct but not representative of the set of possible trajectories.

To simulate a pseudo-random selection of trajectories without requiring extensive modifications to Pysim, a modified version of the model was created, in which the guard condition

[2] `http://stanleybak.com/projects/hycreate/hycreate.html`

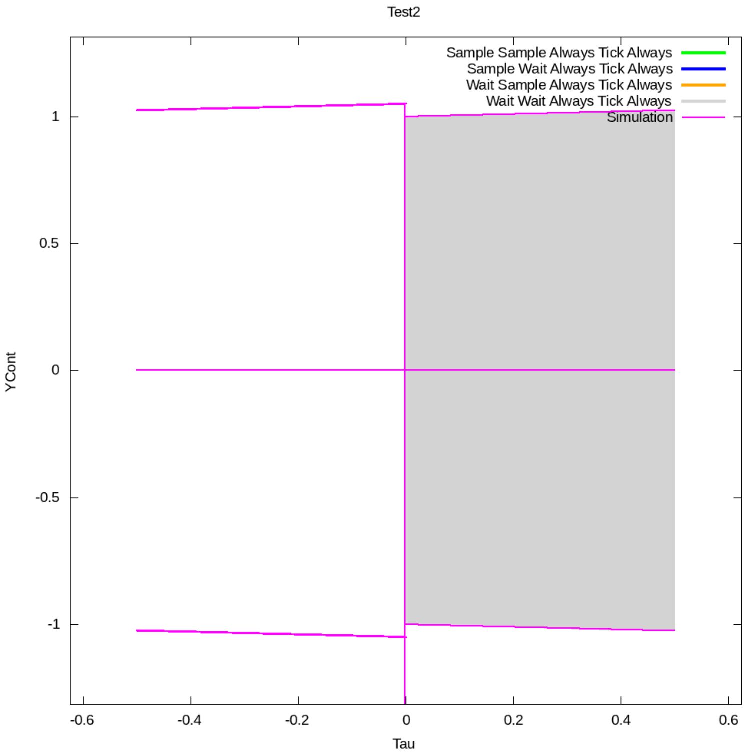

Figure 11: Erroneous reachable set (y over τ) returned by HyCreate: The simulation (lines) goes outside the computed reachable set (shaded area), which does not reach $\tau < 0$. The model file can be found at `code/dummy/hycreate2/test2-converted-fixed.hyc2`.

$t_{min} \leq \tau \leq t_{max}$ in may-semantics was changed to $\tau \geq t_{min} + r(t_{max} - t_{min})$ in must-semantics, where $r \in [0; 1]$ is a pseudo-random number updated at the beginning of each cycle. To fit the existing framework of Pysim and Hyst without extensive modifications, this random update was approximated by

$$r' = 0.5 + 0.5\cos(1234r), \tag{34}$$

which is a prototypical, weak pseudo-random number generator, whose seed value is the initial state of r. Each SH subcomponent has a separate random state, and the initial values are varied to obtain different trajectories.

Additionally, because the equality guard condition $\tau = T/2$ is not correctly simulated due to numerical issues, it was changed to $\tau \geq T/2$, which is equivalent in must-semantics. It remains a topic of future research whether this “model translation” from must- to may-semantics can also be applied for verification with CORA.

C.2 Evaluation of Examples

Due to the results of the previous section, further experiments were restricted to SpaceEx for verification and Pysim for randomized simulation. To handle vector-valued signals of vary-

	n_p	n_d	m	p	timing	SpaceEx	t_SE	K_SE	LTI-stable
A1	1	1	1	1	varying (small)	✓	1 s	1.010	—
A2	1	1	1	1	varying (negligible)	✓	1 s	1.001	—
A3	1	1	1	1	varying (medium)	✓	2 s	1.059	—
A4	1	1	1	1	varying (large)	× error (GLPK)	—	—	—
A5	2	2	2	2	varying (like A3)	× crash (GLPK)	—	—	—
B1	3	2	2	1	varying	✓	16 s	1.097	—
C1	1	2	1	1	constant	× timeout	—	—	stable
C2	1	2	1	1	varying	× crash	—	—	—
D1	3	6	4	3	constant	× diverging	—	—	stable
D2	3	6	4	3	varying	× crash	—	—	—
E	2	1	1	1	varying	—	—	—	unstable

Table 1: Experimental results.
$n_\mathrm{p}, n_\mathrm{d}, m, p$: dimensions of plant, controller, input and output.
timing: fixed or uncertain timing?
SpaceEx, t_SE: result of SpaceEx and runtime (including computation of interval bounds). ✓: practically stable, neglecting floating point inaccuracy. ×: failed to verify practical stability. timeout: runtime exceeded two hours. diverging: reached $K > 1000$ without showing stability. crash: aborted due to unhandled error such as memory access or assertion violation. error: exited with error message. (GLPK: error is related to solving linear programs with the GLPK library)
K_SE: Upper bound of K factor per eq. (11), comparing SpaceEx' overapproximation and random Pysim simulations. Only applicable if SpaceEx verifies stability.
LTI-stability: stability proof by analysis of the time-discretized nominal case ($\Delta t = 0$), which is a linear time-invariant (LTI) system (—: not applicable for varying timing, except to show instability).

ing dimension, which are not directly supported by the SpaceEx file format, the model and configuration files for all examples from appendix B.1 ff. were generated from a template.

Note: Code and data for this section can be found in `code/template/`.

For each example, SpaceEx' maximum number of iterations was adjusted such that either it found a fixpoint, states corresponding to $K > 1000$ were reached or a run-time limit of two hours was exhausted. If a fixpoint was found, this means that the system is practically stable, assuming that floating point computation inaccuracy in SpaceEx is negligible. The remaining settings were chosen as in the initial experiments, except for example B, which had to be changed to `directions=box` (set overapproximation as multidimensional interval) for successful verification. An exhaustive search for the best settings could not be performed in the scope of this work, which means that the presented results may be suboptimal.

The results in table 1 highlight that SpaceEx could verify stability only for the most simple examples (A1 – A3 and B) and fails otherwise.

The results point to three possible reasons for the encountered difficulties.

Firstly, increasing timing variation makes verification more difficult, as can be seen in examples A1 – A4: Example A2 with almost constant timing is verified with almost no bloating, whereas increased timing in A3 leads to more bloating and runtime, and A4 with large timing cannot be verified.

To illustrate the effect of increased timing uncertainty on verification, the reachable set over global time t was computed by adding t as a state with $\dot{t} = 1$ and $t(0) = 0$. Figure 12 highlights that in simulations (left), example A1 (top) shows about the same rate of decay for

$y(t)$ as example A3 (bottom), whereas the reachable set (right) decays significantly slower. This matches the increased K-factor observed in table 1.

Secondly, increasing dimension increases the complexity of verification: While example A5 is simply a duplicated version of A3, verification fails.

Third, the dynamics of the nominal case (perfect timing), such as oscillating versus exponentially decaying, may play an important role: Example C1 has perfect timing and less states than B1, however, only B1 can be verified. While verification with SpaceEx fails for examples C1 and D1, which have constant timing $\Delta t_{\ldots} = 0$, they can easily be proven stable by time-discretization and eigenvalue analysis of the resulting linear time-invariant system.

However, these preliminary interpretations should be taken with a grain of salt, because a number of examples failed with internal errors in SpaceEx whose exact cause could not be determined: Examples A4 and A5 fail with errors related to the solution of linear programs with the GPLK library: A4 causes an error message `GLP_UNDEF`, which suggests numerical issues, and A5 causes an internal error in the GLPK library, for which preliminary research suggests that it has been fixed in newer versions of this library. Examples C2 and D2 fail with segmentation fault (memory access violation) for unknown reasons.

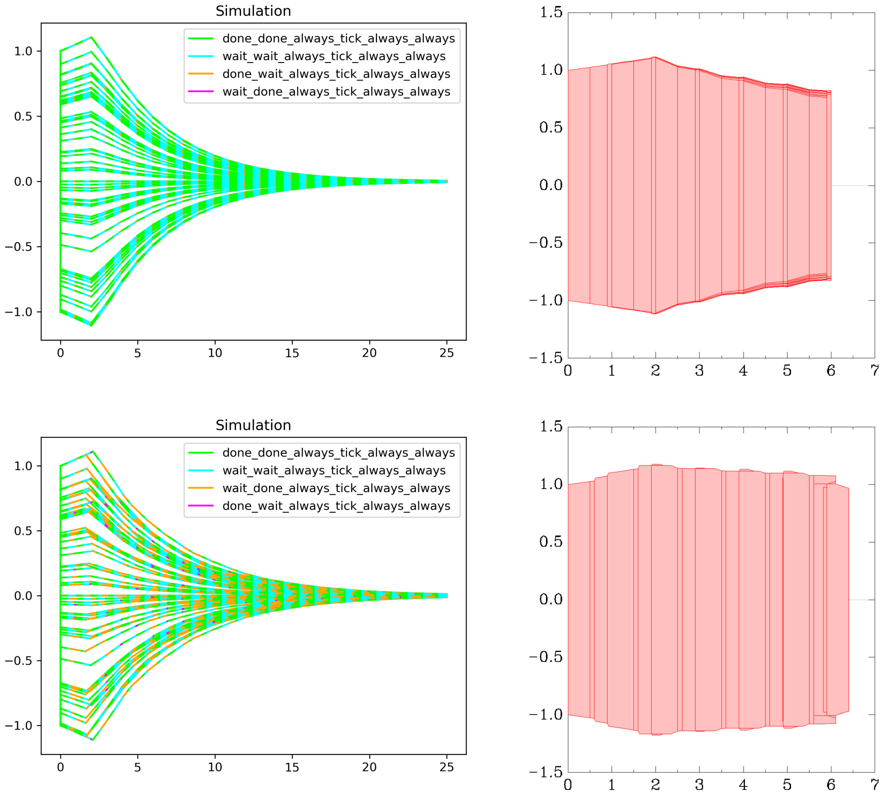

Figure 12: Comparison of random simulations $y(t)$ (left) and the reachable set for y over t computed by SpaceEx (right) for examples A1 (top) and A3 (bottom). Colors in the simulation refer to modes of the hybrid automaton. Input data for this figure can be found in `code/template/output/solved_with_spaceex/A{1,3}_...`.