A New Perspective on Criticality: Efficient State Abstraction and Run-Time Monitoring of Mixed-Criticality Real-Time Control Systems

Tim Rheinfels
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Maximilian Gaukler
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany

Peter Ulbrich
TU Dortmund, Germany

Abstract

The increasing complexity of real-time systems, comprising control tasks interacting with physics and non-control tasks, comes with substantial challenges: meeting various non-functional requirements implies conflicting design goals and a pronounced gap between worst and average-case resource requirements up to the overall timeliness being unverifiable. Mixed-criticality systems (MCS) is a well-known mitigation concept that operates the system in different criticality levels with timing guarantees given only to the subset of critical tasks. However, in many real-world applications, the criticality of control tasks is tied to the system's physical state and control deviation, with safety specifications becoming a crucial design objective. Monitoring the physical state and adapting scheduling is inaccessible to MCS but has been dedicated mainly to control engineering approaches such as self-triggered (model-predictive) control. These, however, are hard to integrate with scheduling or expensive at run-time.

This paper explores the potential of linking both worlds and elevating the physical state to a criticality criterion. We, therefore, propose a dedicated state estimation that can be leveraged as a run-time monitor for criticality mode changes. For this purpose, we develop a highly efficient one-dimensional state abstraction to be computed within the operating system's scheduling. Furthermore, we show how to limit abstraction pessimism by feeding back state measurements robustly. The paper focuses on the control fundamentals and outlines how to leverage this new tool in adaptive scheduling. Our experimental results substantiate the efficiency and applicability of our approach.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Dependable and fault-tolerant systems and networks

Keywords and phrases Real-time Control, Mixed-Criticality, Switched Systems, State Monitoring

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2023.11

Supplementary Material *Software (ECRTS 2023 Artifact Evaluation approved artifact)*: `https://doi.org/10.4230/DARTS.9.1.1`

1 Introduction

For quite some time, we have been facing a rapidly increasing complexity of real-time (control) systems, such as the proliferation of autonomous driving and robotic applications. These are characterized by high performance requirements with numerous control tasks, which interact with physics in closed loops, and non-control tasks executed along with them, forming a heterogenous task set. Meeting all the tasks' non-functional requirements (e.g., QoC, performance, costs) implies conflicting design goals and a pronounced gap between worst and

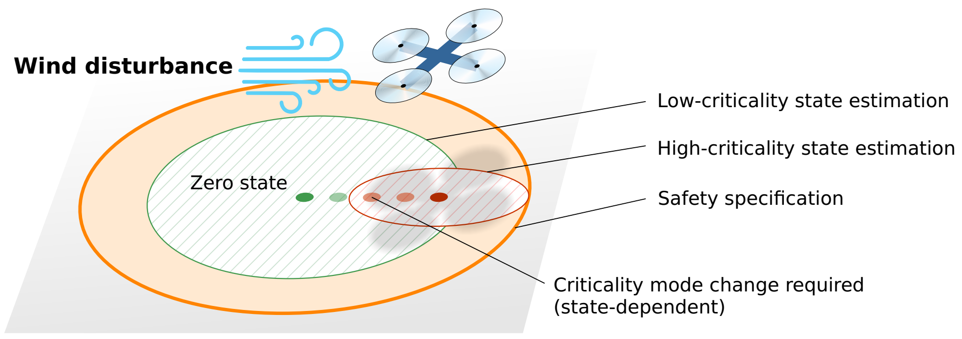
Wind disturbance
Low-criticality state estimation
High-criticality state estimation
Zero state
Safety specification
Criticality mode change required
(state-dependent)

Figure 1 Illustration of a system with safety constraints: the UAV must stay within given bounds (filled area). Compliance is challenged by disturbances (i.e., wind), which, in the worst case, can only be rejected in *hi* mode. Note that criticality change depends on the physical state (i.e., position and velocity vectors). State estimation (hatched areas) varies between modes as laxer scheduling implies more pessimism due to less stringent control.

average-case resource requirements up to the overall timeliness and safety being unverifiable. This gap is a well-known challenge for heterogeneous task sets with a large body of work on mitigation techniques, particularly the concept of mixed-criticality systems (MCS) first introduced by Vestal [42]. It facilitates adapted design and verification of task sets with varying requirements, such that task parameters (e.g., WCETs, periods) become dependent on a criticality level. As a result, MCSs operate in different modes, transparently monitored (e.g., execution time) and enforced by the operating system, with timing guarantees given only to the subset of critical (control) tasks; MCS typically have no intended bearing on the physical side of the system.

As an orthogonal approach, conflicting design goals can be eased by reducing the timing requirements of control tasks based on the inherent robustness of controllers. The latter is exploited to relax, for example, periodicity [11] or deadline adherence [32] while guaranteeing stability. A popular approach is, for example, (m, K)-firm scheduling [21], which requires at least m out of K consecutive job releases to meet their deadlines in terms of a weakly-hard [7] real-time design. Consequently, the controller consistently provides the best possible performance, depending on the utilization, without failing in the worst case [4]. However, this behavior, in turn, represents significant over-provisioning on average. Extended variants of these co-design approaches also allow the controller to adapt to specific job drop scenarios [29] or switch between different controller modes [15], further softening timing constraints. We will discuss a large body of related work in Section 5.

1.1 Problem Statement

In real-world applications, a crucial design objective [2, 6, 38] is to ensure that a system will remain within given safety specifications (e.g., maximum deviation from equilibrium) even at the worst assumed disturbance. Here, the necessary system response's stringency (e.g., temporal) varies with the state-dependent deviation (e.g., control error). Although the control-aware scheduling approaches above can exploit varying demands, they typically aim solely at control stability and thus cannot guarantee compliance with such safety specifications. Note that, in this context, stability only implies that the system reaches equilibrium in the absence of disturbances without any assertions of the largest possible deviation they inflict. On the other hand, a control task's criticality could be characterized by the risk of a safety violation. However, typically, MCS cannot meet such requirements since they do not assess criticality based on control error but on timing, whereby only an indirect relationship exists between temporal and physical properties.

In the following, we use the UAV in Figure 1 as an illustrative example to make our point. It must follow a given flight trajectory, perform collision avoidance, and simultaneously fulfill other mission objectives, all on a shared resource-limited computing platform. In addition, per safety specifications, the UAV must settle and stay within a one-meter radius around its reference (filled area) to avoid crashing into obstacles even in the presence of wind disturbance. On the other hand, higher accuracy inside this envelope is generally desirable yet not mandatory. Finally, assuming worst-case wind disturbances to be a rare failure scenario, we can design the system with a low (lo) and a high (hi) criticality mode. The former facilitates sharing and uses relaxed parameters, for example, larger periods, lax WCETs, or an (m, K) execution model. The latter provides strong isolation, high assurance, and fault rejection. In the given scenario, the maximum wind disturbance drives the UAV off. For some time (i.e., first two states), the relaxed execution behavior in low-criticality mode remains tolerable due to inertia. However, depending on the UAV's physical state, notably its position and velocity vectors, switching to high-criticality mode at a precise moment (i.e., third state) is imperative to avoid the momentum carrying the UAV outside the safety limit. Our concern is that monitoring timing parameters is insufficient to identify the criticality change. Cheng et al. [13] consequently propose using the quality of control as a changeover criterion, however, assuming uncorrelated noise and resorting to a probabilistical model, which is hardly suited for our scenario.

In control engineering, some approaches tie criticality to the current system state. For example, in self-triggered control (STC) methods [23], the controller itself computes the next necessary control instant based on the physical state. Thus, STC releases control tasks sporadically whenever the system threatens to deviate from equilibrium too far, establishing a form of adaptive scheduling. Self-triggered model predictive control (MPC) schemes go even one step further by controlling the process and computing the next sampling instant [25] while enforcing state and control value constraints in the presence of disturbances [28]. However, by unifying state estimation and scheduling, these approaches are inherently incompatible with traditional scheduling methods: jobs must be executed with high timing adherence once released. Thus we lose support for heterogeneous task sets and adaptive scheduling (e.g., criticality-dependent task parameters).

We can address the problem by separating state estimation from controller execution. For the UAV example, a safe decision must be made for a criticality mode. Since the execution conditions in low-criticality mode are more relaxed, the reachable physical state for a given observation period is larger: with less control, the drone can drift further. Conversely, the reachable set is much tighter in high mode. However, robustly observing and predicting the system's physical state is challenging. Established approaches, for example, a set-valued Kalman filter [30], can robustly locate the physical state up to a time-varying ellipsoid. Yet, computational costs have thwarted their use as run-time monitoring for mixed-criticality or other adaptive scheduling approaches at the operating system (OS) level. Our evaluation in Section 4 highlights the overheads associated with set-valued state estimation.

The fundamental issue addressed in this paper is to facilitate the use of the physical state of control systems as a general criticality and scheduling criterion at the OS level. *Our approach offers significant advantages over application-level solutions, such as isolation, standard interfaces, and, most importantly, seamless integration with temporal monitoring*

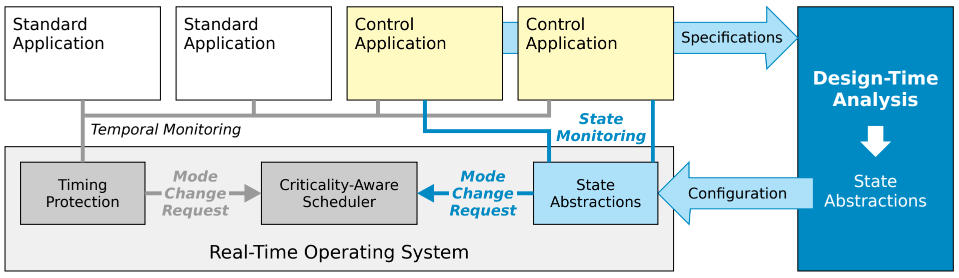

Figure 2 Overview of our approach: implementing state monitoring at the OS level facilitates strong isolation, an application-independent interface, and seamless integration with traditional (MC) execution-time monitoring and scheduling. State abstractions are parameterized by design-time analysis for safe mode switching.

and scheduling of all the system's tasks. As a first step, we focus on low-overhead yet robust (i.e., sound) state estimation at run time and design-time verification of it as a prerequisite for this aim. Therefore, we derive the necessary control engineering background and provide an interface to real-time scheduling. At this point, we emphasize that in this paper, we are concerned with the basic methodology rather than a specific scheduling approach; we believe our approach can serve a wide range of existing scheduling techniques.

Conceptually, we seek a time-dynamic and one-dimensional *state abstraction* v_k:

$$v_{k+1} = \rho_{\sigma_k} v_k + \beta_{\sigma_k}, \quad 0 \leq v_k \overset{!}{\leq} v_{max} \quad \forall k \in \mathbb{N}_0.$$

While the quantity v_k provides an upper bound on the system's state, ρ_{σ_k} denotes the time-varying decay rate (i.e., a lower bound for speed of convergence) and β_{σ_k} the disturbance term (i.e., how the system deviates from equilibrium due to disturbances). Finally, v_{max} gives the application-specific safety requirement (i.e., the maximum permissible deviation from the nominal state). The robust prediction v_{k+1} provides a unified framework to monitor the physical state in the OS-level at run time, for example, allowing us to identify the changeover instant between both modes in our example from Figure 1. However, this seemingly simple requirement is tied to a number of fundamental challenges.

Challenge 1: Run-Time Monitoring by an Easy-to-Compute State Abstraction

Reachability analysis using time-variant sets can yield precise results. However, for example, computing the distance between two sets is prohibitively expensive at run time. Therefore, we aim for a *state abstraction* (i.e., sound estimation) that reliably predicts future violations of the control's safety specification, serving as a run-time monitor for criticality modes. Furthermore, the abstraction should facilitate a simple application-independent interface and simultaneously be economical to compute.

Our Approach. Based on [19], we develop a one-dimensional *state abstraction* (called blind abstraction) for switched linear control systems. This allows us to restrict the prediction horizon to one time step (e.g., time slice of the scheduler), permitting a timely mode change and, if necessary, an adaptation of the control regime to reject worst-case disturbance safely. Furthermore, reducing the state dimensions and prediction horizon grants low run-time overhead and a simple interface. Figure 2 outlines our approach and illustrates how the state abstractions fit the OS kernel and scheduling.

Challenge 2: Selective Reduction of State Abstraction Overestimation

Abstraction reduces complexity but is typically accompanied by considerable pessimism. Accordingly, we must ensure appropriate accuracy by leveraging the available state information. Furthermore, executing modes of lower criticality causes the state information to become increasingly uncertain. Consequently, the abstraction's overapproximation could lead to premature anticipation of safety violations causing false changeovers. Such behavior would jeopardize its use as a monitoring function.

Our Approach. We introduce the concept of an *observer abstraction*, which mitigates the statically inferred worst-case pessimism of the abstraction by robustly feeding uncertain measurements of the system state into the (formerly blind) abstraction.

Challenge 3: Design-Time Analysis for Efficiently Choosing the Parameters

Finally, we must ensure that all abstractions fit the system and are safe to use. The challenge is to tailor our abstractions at design time to a system with multiple criticality levels and system parameters for which potentially only the most stringent one guarantees adherence to the safety specification. Simply put, starting with the worst-case observer abstraction for the worst case, we must parameterize the observers for the lower criticality levels such that the system remains in optimistic execution with high probability and that each criticality level has a reasonable operating range.

Our Approach. Based on semidefinite programming, we develop a heuristic which optimizes all design-time parameters for the average case while still guaranteeing safety in the worst case. This analysis, as shown in Figure 2, is used to parameterize the state abstractions offline based on the control applications' specification and criticality modes.

1.2 Contribution and Outline

This paper makes the following four contributions: (1) Extend the convergence rate abstractions presented in our previous work [19] to a more general linear system model. (2) An observer-based approach to mitigate over-estimation in those state abstractions by feeding back uncertain measurements robustly. (3) Determine safe change-over points for criticality and implement a prototypical adaptive run-time policy. (4) A design-time heuristic to parameterize the observer abstractions' parameters.

The remainder is organized as follows. In Section 2, we detail our system model and present relevant background information. Our approach on efficient and robust state monitoring to facilitate run-time adaptivity is presented informally as well as formally in Section 3 while the necessary mathematical proofs are detached to Appendix A. Section 4 provides experimental results as part of a case study. Section 5 discusses related work. Finally, we conclude our work in Section 6.

2 System Model

In this section, we define the paper's notation, describe the mathematical foundations, and formulate our control-theoretic system model. Further, we outline integrating the latter into a real-time system's scheduling and schedulability analysis.

2.1 Notation, Ellipsoids, and Relevant Norms

We denote both scalars $\rho \in \mathbb{R}$ and vectors $x \in \mathbb{R}^n$ as lowercase, matrices $A \in \mathbb{R}^{n\times n}$ as uppercase letters. The identity matrix of appropriate size is referred to as I. We use A^T and A^{-1} for a matrix's transposed and inverse and denote a symmetric and positive definite (s.p.d.) / semidefinite (s.p.sd.) matrix $P = P^T$, $x^T P x > 0 \forall x \neq 0$ / $x^T P x \geq 0 \forall x$ as $P \succ 0$ / $P \succeq 0$. For an s.p.d. matrix $P \succ 0$, we refer to its lower Cholesky decomposition as $P^{\frac{1}{2}}$, i.e. $P = P^{\frac{1}{2}} P^{\frac{T}{2}}$. Given two symmetric matrices $P = P^T, Q = Q^T$, we abbreviate $x^T P x \geq x^T Q x \ \forall x$ as $P \succeq Q$.

Based upon the Euclidean vector norm $||x||$, we define the P-weighted norm $||x||_P := ||P^{\frac{T}{2}} x||$ with $P \succ 0$. The inequality $||x||_P \leq 1$ defines a non-degenerate and centered ellipsoid.

Given the spectral norm $|| \cdot ||$, we generalize the vector norms $|| \cdot ||_P$ to matrix norms $||A||_{PQ}$ in terms of the s.p.d. weight matrices $P, Q \succ 0, P \in \mathbb{R}^{n\times n}, Q \in \mathbb{R}^{m\times m}$. With correctly sized x and s.p.d. $W \succ 0$, the following hold [46, pp. 34–35]:

$$||A||_{PQ} := \max_{||x||_Q = 1} ||Ax||_P \qquad \textit{(Definition as Operator Norm)} \qquad (1a)$$

$$||A||_{PQ} = ||P^{\frac{T}{2}} A Q^{-\frac{T}{2}}|| \qquad \textit{(Relation to Spectral Norm)} \qquad (1b)$$

$$||Ax||_P \leq ||A||_{PQ} ||x||_Q \qquad \textit{(Consistency)} \qquad (1c)$$

$$||AB||_{PQ} \leq ||A||_{PW} ||B||_{WQ} \qquad \textit{(Generalized Consistency)} \qquad (1d)$$

The last equation (1d) is a generalization of [46] (Wang et al. assume P=Q). Given the sub-multiplicativity ($||AB|| \leq ||A|| \, ||B||$) of the spectral norm, the proof is:

$$||AB||_{PQ} \overset{(1b)}{=} ||P^{\frac{T}{2}} A \underbrace{W^{-\frac{T}{2}} W^{\frac{T}{2}}}_{=I} B Q^{-\frac{T}{2}}|| \leq ||P^{\frac{T}{2}} A W^{-\frac{T}{2}}|| \, ||W^{\frac{T}{2}} B Q^{-\frac{T}{2}}|| \overset{(1b)}{=} ||A||_{PW} ||B||_{WQ}. \quad (2)$$

Further, we will be using the defining properties of (semi-) norms, i.e., for any norm $|| \cdot ||$, vectors x, y, and scalar α

$$||\alpha x|| = |\alpha| \, ||x|| \qquad \textit{(Homogeneity)} \qquad (3a)$$

$$||x|| \geq 0 \qquad \textit{(Non-negativity)} \qquad (3b)$$

$$||x + y|| \leq ||x|| + ||y|| \qquad \textit{(Triangle Inequality)} \qquad (3c)$$

The notation $a \overset{!}{\leq} b$ indicates that we must ensure $a \leq b$ by applying adequate measures.

2.2 System Model

This section defines the system model used in the remainder of the paper. For each control application, we assume a switched linear plant and controller with a finite number of modes. All states, potentially including sensor and actuator values, are summarized into an extended linear system. This model allows for complex real-time behavior, e.g., omitting controller executions or sampling only a part of the available sensors [45, 44, 18]. Adding safety outputs allows us to both disregard states (e.g., controller states or measurements) aiding the analysis or impose additional constraints (e.g., limits on the control signals). The safety goal is to keep these outputs inside of a given ellipsoid. The dynamics are influenced by an ellipsoidally bounded process uncertainty, as are the uncertain measurements. The initial state is also constrained to an ellipsoid. The state-space model reads

$$\begin{cases} x_{k+1} = A_{\sigma_k} x_k + G_{\sigma_k} d_k & \textit{(Dynamics)} \quad (4a)\\ y_k = C_{\sigma_k} x_k + H_{\sigma_k} z_k & \textit{(Measurements)} \quad (4b)\\ ||d_k||_{D_{\sigma_k}} \le 1,\ ||z_k||_{Z_{\sigma_k}} \le 1 & \textit{(Disturbances)} \quad (4c)\\ ||x_0||_{X_0} \le 1 & \textit{(Initial State)} \quad (4d)\\ s_k := C_s x_k & \textit{(Safety Outputs)} \quad (4e)\\ ||s_k||_S \overset{!}{\le} 1 & \textit{(Safety Specification)} \quad (4f)\\ \sigma_k \in \Sigma,\ |\Sigma| = n_\Sigma & \textit{(Switching Sequence)} \quad (4g) \end{cases}$$

$\forall k \in \mathbb{N}_0$, where $x_k \in \mathbb{R}^{n_x}$, $d_k \in \mathbb{R}^{n_d(\sigma_k)}$, $z_k \in \mathbb{R}^{n_z(\sigma_k)}$, $y_k \in \mathbb{R}^{n_y(\sigma_k)}$, and $s_k \in \mathbb{R}^{n_s}$ are the system state, process and measurement disturbances, as well as the measurements and the safety outputs. All matrices should be of appropriate (possibly zero) size. The ellipsoids (if not of zero dimension) are assumed to be non-degenerate, i.e. D_σ, Z_σ, X_0, and $S \succ 0\ \forall\sigma \in \Sigma$.

Given the system model, we define the *worst-case disturbance* as the values of x_0, d_k and z_k, which steer the safety outputs the closest to the specification boundary for a given switching sequence σ_k, i.e., $\sup_{x_0 \in X_0, d_k \in D_{\sigma_k}, z_k \in Z_{\sigma_k}, l \in \mathbb{N}_0} ||s_l||_S$. These values can not readily be obtained as the optimization horizon is infinite.

For the sake of clarity, we omitted any reference signals $w_k \in \mathbb{R}^{n_x}$, $x_{r,k+1} = A_{\sigma_k} x_{r,k} + w_k$ and $y_{r,k} = C_{\sigma_k} x_{r,k}$. They can, however, be incorporated by substituting x_k and y_k in the system above by their corresponding error signals $\Delta x_k := x_k - x_{r,k}$ and $\Delta y_k := y_k - y_{r,k}$. The abstraction then bounds the tracking error, which leads to the same analytical results as below due to the system's linearity.

We assume that, except for potential reference signals, a single application implements the above controller. Every criticality mode σ inside this application has its own task set. Our approach is agnostic to the underlying scheduling scheme as long as the signals meet their sampling points. This can, e.g., be achieved using the logical execution time (LET) paradigm. Multiple independent control applications can be used if the scheduling algorithm guarantees their timeliness.

3 Approach

With our system model defined, we proceed by detailing our state abstraction approach in this section. In an informal description in Section 3.1, we first recapitulate the intent of state abstractions, describe how they interact with the real-time control system, and discuss what sets them apart from well-known state observers. After Section 3.2 extends the convergence rate abstractions presented in our previous work [19] to our system model, Section 3.3 shows how to incorporate measurements to reduce their pessimism. In Section 3.4, we derive the run-time monitoring algorithm and a prototypical approach for guaranteed safe switching decisions. Finally, Section 3.5 describes a heuristic for choosing the design-time parameters. To better convey the ideas, we detach the necessary formal proofs to Appendix A and reference them as needed.

3.1 Informal Description

We seek a method of monitoring the system model presented in Section 2.2 at run time with respect to safety, i.e., how close the physical state is to its specified limits and, in extension, predict points where criticality changes. When reaching such a point, the scheduler must alter the switching sequence σ_k to avoid specification violations. Note that safety is a stronger condition than mere exponential stability for which the literature already provides solutions [45].

Our approach provides *(state) abstractions* v_k, which are one-dimensional and positive dynamic systems. An abstraction's state v_k is an upper bound on the control system's physical state expressed in the *analysis norm* $|| \cdot ||_P$, i.e., $0 \le ||x_k||_P \le v_k \forall k \in \mathbb{N}_0$. As a geometric interpretation: instead of the ellipsoidal observers, which track how the system's dynamics recursively alters the initial state ellipsoid at run time, we choose a fix-shaped analysis ellipsoid parametrized by $P \succ 0$ at design time, leaving the scaling factor v_k as the only run-time parameter. The abstraction dynamics, i.e., the coefficients ρ_σ and β_σ mentioned above, are chosen such that the scaled analysis ellipsoid is guaranteed to contain the system's state for the switching sequence σ_k at hand. Tightly over-approximating the initial state and safety ellipsoids yields the two static factors v_0 and v_{max}. While the former is the initial value for the abstraction state, the latter links it to the safety goal: for our approach to guarantee safety, $v_k \le v_{max}$ must be enforced at all times by choosing the switching sequence accordingly.

The intermediate steps introduce analysis pessimism as the ellipsoids are usually not aligned in practice. Further, the exponential envelope curve v_k has to incorporate all transient behavior and overshoot in the system state even under worst-case disturbance. Despite the high pessimism, the lack of overshoot allows us to derive scheduling decisions at a single time step prediction horizon. We will later introduce *observer abstractions* to alleviate the pessimism by feeding back uncertain state measurements. The analysis presented in this section expresses the scheduling horizon in terms of a *maximum safe value* v_σ^* for every mode $\sigma \in \Sigma$, i.e., the highest value of v_k for which the corresponding mode is sufficient. Consequentially, modes with higher v_σ^* indicate higher physical criticality as they allow the control loop to withstand more severe conditions. If $v_\sigma^* \ge v_{max}$ holds for a mode σ, it is a *safe mode* as applying it guarantees the system to remain safe at all times. We call the excess $v_\sigma^* - v_{max}$ the *safety margin.*

While set-valued observers are available for our system model [16, 30], they perform the aforementioned ellipsoidal analysis at run time, yielding a precise bound on the system's state. However, choosing the switching sequence σ_k requires further assessing the system's reachability at run time which is expensive as the ellipsoids are not aligned in general. In contrast, their non-robust counter parts (e.g., the classical Luenberger observer) give static guarantees on the estimation error's decay. Going down this path will eventually lead to similar results as presented in this paper.[1]

Since our analysis is sound as long as the IO timing is met, every (mixed-criticality) scheduling scheme suffices when choosing the control application's criticality high enough. Conversely, the abstraction introduces some overhead. The value of v_k has to be updated every time step given (8a), potentially reusing the measurements from the controller. Then, the scheduler has to select one of the feasible modes defined by (13) for the upcoming time step and possibly reconfigure the real-time system accordingly using a mode change. These operations look the same for every control loop and thus share a common interface. As shown in Section 4, they are also cheap to compute. Thus, they lend themselves to be executed as privileged operations inside the RTOS's kernel.

[1] We omitted the analysis for the sake of brevity. As an outline: compute the estimate $\hat{x}_k$ as usual and bound the uncertainty ellipsoid just as the blind abstractions bound the state. Then, $||\hat{x}_k||_P - v_k \le ||x_k||_P \le ||\hat{x}_k||_P + v_k \forall k$.

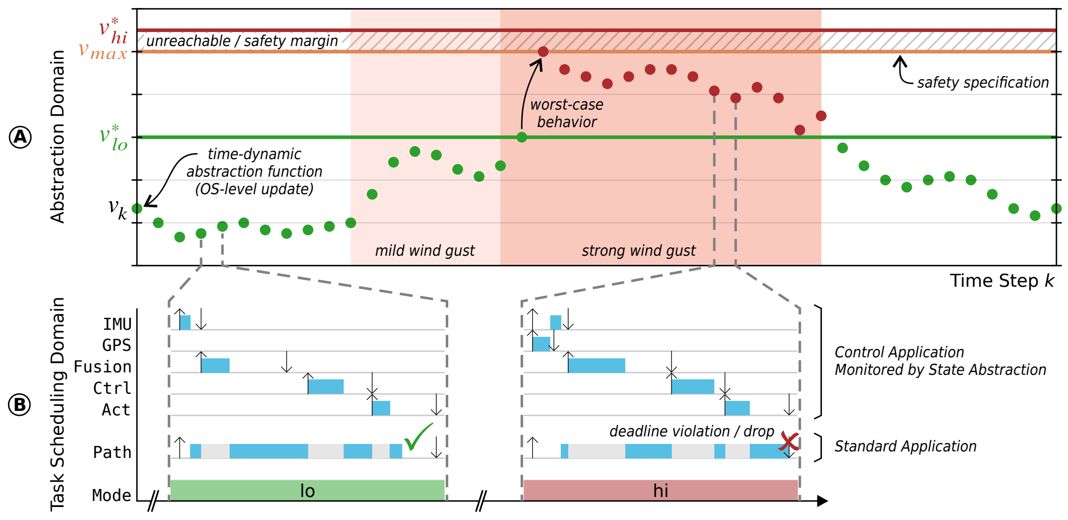

Figure 3 Example task set illustrating the application of state abstractions for mixed-criticality real-time control systems. (A) shows the time-dynamic state abstraction monitoring a UAV under the influence of varying wind conditions. (B) illustrates the resulting schedule with criticality change.

We exemplify our approach using the UAV from above as an illustrative example. The safety goal is to keep the UAV's position p_k inside a one-meter radius around its reference trajectory. Thus, the safety outputs omit all other state components such as the UAV's velocities, i.e., $s_k = p_k$. $S = I$ models the 1m radius. Figure 3 (B) shows the exemplary task set: the flight controller application carries out the state reconstruction by linearly fusing inertial (IMU) and GPS measurements, then computes the control values, and finally outputs them to the propellers. To avoid jitter, sensing and actuation follow the logical execution time (LET) paradigm. The application features two periodic task sets whose physical (i.e., abstraction coefficients) and temporal properties vary with their criticality levels: in the lo-criticality mode $\sigma = 1$, the IMU measurement task is omitted, also leading to a smaller WCET for the sensor fusion task in comparison to the hi-criticality mode $\sigma = 0$. Consequently, the deadline for the controller task can be relaxed, leaving enough resources to guarantee schedulability of the less critical path-planning application. The latter comprises only a single long-running periodic task altering the controller's set point.

At the beginning of the timeline in Figure 3 (A), the UAV is in good condition, indicated by $v_k \leq v^*_1$. Measurements allow the observer abstraction to detect the mild gust of wind and its state rises in turn. While the lo-criticality mode $\sigma = 1$ can reject this disturbance, the strong gust of wind occurring later drives the UAV further off its reference trajectory, eventually leading to a change in criticality when v_k passes v^*_1. Here, the scheduler timely reconfigures the system for the hi-criticality mode, which can withstand the strongest specified disturbance for any amount of time. However, the increased computational demand for $\sigma = 0$ implies that guaranteed timeliness for the less critical path planning application must be dropped. Figure 3 (B) illustrates how worst-case timing leads to a deadline violation in the latter. Recapitulating, this signifies the notion of physical mixed-criticality introduced by our approach: in the average case, the whole system works as expected. However, during unusually high disturbances, the criticality changes based on the underlying physical process. If the system becomes overloaded, it still has a defined fault state. In the UAV's case, it will hover around the last known reference point until the disturbance wears off and then continue its flight.

3.2 Blind Abstractions

As a first step, we apply the idea presented in [19] to our system model (4) by restricting it to the linear case and extending it to support ellipsoidal bounds and multiple switching modes. For this, we base the analysis on the $||\cdot||_{PQ}$-norms introduced in Section 2.1.

While the rigorous proofs are detached to Appendix A, we still want to exemplify the derivation here. The key mathematical idea is to express the system's safety outputs under the $||\cdot||_S$-norm, split the resulting equation by applying the triangle inequality, and upper bound the uncertain terms by their specified ellipsoids. Introducing an analysis ellipsoid parametrized by the matrix $P \succ 0$ yields a time-dynamic expression $||x_k||_P$, which we can relate to the specification. Applying the ideas, we get

$$||s_{k+1}||_S \overset{(4a),(4e)}{=} ||C_s(A_{\sigma_k}x_k+G_{\sigma_k}d_k)||_S \overset{(3)}{\leq} ||C_s||_{SP}(||A_{\sigma_k}||_{PP}||x_k||_P+||G_{\sigma_k}||_{PD_{\sigma_k}}) \overset{(4f),!}{\leq} 1. \quad (5)$$

Appendix A gives a more formal derivation. There, we prove why the following linear system, which we call *blind (state) abstraction*, provides a sound upper bound (Theorem 3) on the system's state, i.e., $||x_k||_P \leq v_k \ \forall k \in \mathbb{N}_0$ and how obeying the specification translates to a constant-valued *safety bound* v_{max} on the abstraction's state (6c) (Theorem 4). $\forall k \in \mathbb{N}_0$:

$$\begin{cases} v_{k+1} = \rho_{\sigma_k} v_k + \beta_{\sigma_k} & \textit{(Dynamics)} \quad (6a) \\ v_0 = \alpha & \textit{(Initial State)} \quad (6b) \\ v_k \overset{!}{\leq} v_{max}. & \textit{(Safety Bound)} \quad (6c) \end{cases}$$

All coefficients can be derived from the system model (4) by applying different $||\cdot||_{PQ}$-norms as

$$\rho_\sigma := ||A_\sigma||_{PP} \quad \beta_\sigma := ||G_\sigma||_{PD_\sigma} \quad v_{max} := ||C_s||_{SP}^{-1} \quad \alpha := ||I||_{PX_0}. \quad (7)$$

While α is the abstraction's lowest admissible initial state, ρ_σ and β_σ define the worst-case exponential decays and disturbance influences for each mode $\sigma \in \Sigma$, respectively. Aside from being *one-dimensional*, the abstraction's state v_k inherits the desirable property of non-negativity from the norm-based coefficients. In combination, this allows for easy monitoring of the system with respect to two values with clear interpretations: while for $v_k = 0$ the system is in perfect condition, $v_k = v_{max}$ indicates that the abstraction is on the verge of guaranteeing safety.

3.3 Adding Measurements: Observer Abstractions

We argue that the blind abstraction (6) poses a valuable analysis tool. However, as we will show in Section 4, the pessimistic assumptions make it hard to use in practice. This section presents how uncertain measurements can be fed into the abstraction to improve the average-case behavior. Casually speaking, the measurement allows the arising *observer abstractions* to rule out impossible abstraction values. The underlying assumption is that these impossible cases are more often on the pessimistic side than not, improving the abstraction's performance on average.

The key concept behind incorporating measurements is to add a Luenberger observer, i.e., superimposing the state estimate with some linear combination of the measurements. The observer gains $L_\sigma \in \mathbb{R}^{n_x \times n_y(\sigma)}$ define the weights for every state and measurement component. As shown by Lemma 1, the observer abstractions truly generalize the blind ones when disregarding the measurements, i.e., choosing $L_\sigma = 0\ \forall \sigma \in \Sigma$. In a slight abuse of notation, we redefine the symbols used before and arrive at the following definition of *observer abstractions.* $\forall k \in \mathbb{N}_0$:

$$\begin{cases} v_{k+1} = \rho_{\sigma_k} v_k + \beta_{\sigma_k} + ||L_{\sigma_k} y_k||_P & \textit{(Dynamics)} \quad (8a) \\ v_0 = \alpha & \textit{(Initial State)} \quad (8b) \\ v_k \overset{!}{\leq} v_{max}. & \textit{(Safety Bound)} \quad (8c) \end{cases}$$

Comparing (6) and (8), their structure differs only in the additional measurement term in the dynamics equation. To discuss how adding measurements alters the abstraction, we first introduce the observer dynamics matrix to abbreviate the results later on:

$$\tilde{A}_\sigma := A_\sigma - L_\sigma C_\sigma. \tag{9}$$

With this in mind, the coefficients for (8) are again $||\cdot||_{PQ}$-norms of the matrices defining the system model (4). They read

$$\begin{cases} \rho_\sigma := ||\tilde{A}_\sigma||_{PP} \quad \beta_\sigma := ||G_\sigma||_{PD_\sigma} + ||L_\sigma H_\sigma||_{PZ_\sigma} \quad v_{max} := ||C_s||_{SP}^{-1} \quad \alpha := ||I||_{PX_0} & (10a) \\ \gamma_\sigma := ||L_\sigma C_\sigma||_{PP} \quad \delta_\sigma := ||L_\sigma H_\sigma||_{PZ_\sigma}. & (10b) \end{cases}$$

Besides the coefficients already introduced for the blind abstractions, assessing the worst-case behavior of the observer abstractions requires the two additional terms γ_σ and δ_σ. Lemma 5 shows that the dynamic system

$$\begin{cases} \bar{v}_{k+1} = (\rho_{\sigma_k} + \gamma_{\sigma_k})\bar{v}_k + \beta_{\sigma_k} + \delta_{\sigma_k} \quad \forall k \in \mathbb{N}_0 & (11a) \\ \bar{v}_0 = \alpha & (11b) \end{cases}$$

provides an upper bound on the observer abstraction, i.e., $v_k \leq \bar{v}_k\ \forall k \in \mathbb{N}_0$, effectively reducing the worst-case behavior to a more pessimistic version of the blind abstraction (6). Interpreting this result, the coefficients γ_σ and δ_σ describe the *additional* decay rate and disturbance input separating the average from the worst case.

Assuming that the pair (A_σ, C_σ) is sufficiently observable,[2] the gains L_σ allow for altering the abstraction dynamics for the better (i.e., reduce ρ_σ and β_σ) on average. This reduction comes at the cost of adding uncertainty originating from the measurements via δ_σ. We want to make clear that we are still concerned with how the system states behave, not the observer's error dynamics. It is impossible to alter the system's dynamic properties via the measurements, even if they are perfect, i.e., $\delta_\sigma = 0$. They only allow for improving upon our knowledge about its state. The triangle inequality reflects this mathematically:

$$||A_\sigma||_{PP} \overset{(9)}{=} ||\tilde{A}_\sigma + L_\sigma C_\sigma||_{PP} \overset{(3c),(10)}{\leq} \rho_\sigma + \gamma_\sigma.$$

As a note, the observer gains shown here are independent of any observer encapsulated in (4). While they allow for adding any measurement of the form (4b), we assume that the abstraction is only fed a subset of those measurements used for the controller as querying additional sensors at run time defies the goal of cheap state monitoring.

[2] While full observability allows for choosing ρ_σ arbitrarily small or even as 0 [33, 40], its value can already be reduced with only subsystems being observable.

3.4 Safe Abstraction-Based Run-Time Switching Policies

If we recapitulate on the previous sections, we have found a safe upper bound for assessing the system's state and how well it obeys its safety specification by introducing state abstractions (8). Based upon the assumption of a *safe mode*, i.e., one which can provably withstand the worst-case disturbance over any amount of time, we derive the set of feasible switching modes for the current abstraction state. Finally, we describe a prototypical algorithm for abstraction-based run-time scheduling.

To find the set of feasible modes and thus determine points at which criticality changes, we first use the results from Lemma 5 to define the *maximum safe value* v^*_σ for every mode σ as the highest abstraction value which guarantees that the abstraction stays within its safety bound for the next time step when choosing $\sigma_k = \sigma$:

$$v^*_\sigma := \frac{v_{max} - \beta_\sigma - \delta_\sigma}{\rho_\sigma + \gamma_\sigma} \quad \forall \sigma \in \Sigma. \tag{12}$$

Note that modes with negative v^*_σ are infeasible, i.e., our analysis provides them with no safety guarantees at any time. In contrast, the *feasible set* $\Sigma_f(v_k)$ at time step k defines all modes for which the abstraction stays safe for $k+1$, i.e.,

$$\Sigma_f(v_k) := \{\sigma \in \Sigma | v_k \leq v^*_\sigma\}. \tag{13}$$

By Theorem 7, this set is never empty if a) the system is feasible and b) there is at least one *safe mode* σ permitting worst-case disturbance for any amount of time, which translates to

$$v_0 \leq v_{max} \quad \text{and} \tag{14}$$

$$\exists \sigma \in \Sigma : \quad v^*_\sigma \geq v_{max}. \tag{15}$$

Notice that the condition (15) requires exponential stability, i.e., $\rho_\sigma + \gamma_\sigma < 1$. If these assumptions hold, the system is guaranteed to remain safe by Corollary 8 when the scheduler chooses the switching sequence from the feasible set (13):

$$\sigma_k \in \Sigma_f(v_k) \quad \forall k \in \mathbb{N}_0 \Rightarrow ||s_k||_S \leq 1 \quad \forall k \in \mathbb{N}_0. \tag{16}$$

We now define a *prototypical switching policy.* It greedily chooses the least critical but feasible mode σ for the timestep k by assessing the abstraction's state v_k. Without loss of generality, we assume that the states are ordered by criticality with $\sigma = 0$ being the highest critical one. Formally:

$$\sigma_k = \max \Sigma_f(v_k) \quad \forall k \in \mathbb{N}_0. \tag{17}$$

3.5 Design-Time Analysis

Aside from the analysis ellipsoid parametrized by the matrix $P \succ 0$, each set of observer gains L_σ introduces additional parameters. While the problem of choosing them looks similar to designing a robust observer for switched systems for which the literature provides optimal solutions [16], we are again not concerned with the observer's error dynamics but the system's safety. Therefore, the problem is more challenging: as shown in Section 3.3, adding an observer will – at best – not increase the abstraction coefficients (10). However, as of Section 3.4, we need only one safe mode to guarantee safety at all times. This allows us to adapt the parameters accordingly and incorporate the other (possibly unstable) modes to best effort. In this section, we derive a *heuristic* for choosing the parameters for large v^*_σ by formulating a parametric semidefinite program (SDP) and solving it on a grid $i \in \mathcal{G}$. To keep the parameters and variables for the individual optimization problems distinct from the derivations above, we index them using i as, e.g., $P(i)$. Afterward, we choose the actual abstraction parameters as the best feasible solution found on the grid. Note that this part of the paper is not a rigorous derivation but a piece of engineering that works for our purposes. Consequently, we focus on conveying the underlying thoughts instead of detailing the fundamentals of and modeling with semidefinite programming. We refer the interested reader to e.g., [41] and [16].

As a starting point, the abstraction coefficients (10) can be computed for every $P \succ 0$. This allows us to apply any heuristic for choosing P and the L_σ and later assess if the resulting abstraction guarantees safety. With that in mind and for the sake of brevity, we do not give proofs for the derivation of the heuristic and formulate assumptions from which we proceed. Informally, the SDP aims to maximize the smallest $v^*_\sigma(i)$, i.e., make the least critical mode schedulable for the longest amount of time. In the following, we will detail the optimization problem mathematically. Most of the constraints are needed to express different $||\cdot||_{PQ}$-norms, but we also add additional ones to enforce safe modes.

The first set of constraints follows from the static portions of the system model (4): we require that the safety outputs obey the specification for all initial states, i.e., $||x_0||_{X_0} \leq 1 \Rightarrow ||C_s x_0||_S \leq 1$. The following linear matrix inequality (LMI) is mathematically equivalent:

$$X_0 \overset{!}{\succeq} C_s^T S C_s. \tag{18}$$

We want to ensure that the analysis ellipsoid of the i-th problem expressed by the matrix variable $P(i) \succ 0$ is non-degenerate and well-conditioned. For convenience, we confine it between the initial state and the specification ellipsoid, leading to $0 \leq v_0 \leq v_{max} \leq 1$ if the problem is feasible. To avoid a loss of definiteness in case of C_s not being full-ranked, we restrict the smallest eigenvalue to the arbitrary value $\sqrt{10^{-6}}$. The following LMIs express all of this:

$$P(i) \overset{!}{\succeq} C_s^T S C_s \quad \text{and} \quad X_0 \overset{!}{\succeq} P(i), \quad P(i) \overset{!}{\succeq} 10^{-6} \cdot I. \tag{19}$$

To make the optimizer aware of the $v^*_\sigma(i)$, we must provide it with the abstraction coefficients (10), which depend on $P(i)$. We assume there is no convex representation for choosing the $\rho_\sigma(i)$, $\gamma_\sigma(i)$, $L_\sigma(i)$, and $P(i)$ simultaneously, use the former two as parameters for the semidefinite program, and confine them to a grid later. To incorporate the observer gains, we define the optimization variables $W_\sigma(i) := P(i)L_\sigma(i)$. By Schur's complement and after multiplying with $P(i)$, we can rewrite $||\tilde{A}_\sigma(i)||_{P(i)P(i)} \leq \rho_\sigma(i)$ and $||L_\sigma(i)C_\sigma||_{P(i)P(i)} \leq \gamma_\sigma(i)$ as $\forall \sigma \in \Sigma$:

$$\begin{bmatrix} \rho_\sigma(i)P(i) & (P(i)A_\sigma - W_\sigma(i)C_\sigma)^T \\ P(i)A_\sigma - W_\sigma(i)C_\sigma & \rho_\sigma(i)P(i) \end{bmatrix} \overset{!}{\succeq} 0 \text{ and } \begin{bmatrix} \gamma_\sigma(i)P(i) & (W_\sigma(i)C_\sigma)^T \\ W_\sigma(i)C_\sigma & \gamma_\sigma(i)P(i) \end{bmatrix} \overset{!}{\succeq} 0. \tag{20}$$

We want to avoid introducing additional parameters and, therefore, an exponential increase in the search space they span. Thus, we define the optimization variables $\beta_k^2(i)$ and $\delta_k^2(i)$ and (probably falsely) assume that $\beta_k^2(i) \approx \beta_k$ and $\delta_k^2(i) \approx \delta_k$. This makeshift is valuable as we can avoid non-convex products such as $\delta_\sigma(i)P^{-1}(i)$ and instead reformulate $||G_\sigma||^2_{P(i)D_\sigma} \leq \beta^2_\sigma(i)$ and $||L_\sigma(i)H_\sigma||^2_{P(i)Z_\sigma} \leq \delta^2_\sigma(i)$ as the LMI contraints

$$\begin{bmatrix} \beta^2_\sigma(i)D_\sigma & (P(i)G_\sigma)^T \\ P(i)G_\sigma & P(i) \end{bmatrix} \overset{!}{\succeq} 0 \quad \text{and} \quad \begin{bmatrix} \delta^2_\sigma(i)Z_\sigma & (W_\sigma(i)H_\sigma)^T \\ W_\sigma(i)H_\sigma & P(i) \end{bmatrix} \overset{!}{\succeq} 0 \quad \forall \sigma \in \Sigma. \tag{21}$$

To express the maximum safe values, we rewrite (12) as $(\rho_\sigma + \gamma_\sigma) v^*_\sigma + \beta_\sigma + \delta_\sigma \leq v_{max}$ and introduce $v^*_\sigma(i)$ as optimization variables. We have found empirically that this inequality can be approximated as $c_\sigma(i) := (\rho_\sigma(i) + \gamma_\sigma(i)) v^*_\sigma(i) + \beta^2_\sigma(i) + 4\delta^2_\sigma(i) \leq v_{max}$. Given the norm-based definition (10) of v_{max}, which is also used to obtain its actual value after optimization, we get

$$\begin{bmatrix} P(i) & c_\sigma(i) C_s^T \\ c_\sigma(i) C_s & S^{-1} \end{bmatrix} \overset{!}{\geq} 0 \quad \forall \sigma \in \Sigma. \tag{22}$$

We require a subset of modes $\emptyset \neq \Sigma_s \subseteq \Sigma$ to be safe which translates to

$$v^*_\sigma \overset{!}{\geq} v_{max} \quad \forall \sigma \in \Sigma_s. \tag{23}$$

As a last set of constraints, we introduce the single variable $v^*_{min}(i)$ as the minimum of all $v^*_\sigma(i)$,

$$v^*_{min}(i) \overset{!}{\leq} v^*_\sigma(i) \quad \forall \sigma \in \Sigma, \tag{24}$$

and use it as the maximization objective. Requirements on the modes' exponential decays can then be imposed using the optimization parameters $\rho_\sigma(i)$ and $\gamma_\sigma(i)$. The next step is to define a grid for their values on which to solve the SDP. The parameter space grows exponentially in the number of modes. We consider this to be undesired and consequentially constrain all values $\rho_\sigma(i)$ and $\gamma_\sigma(i)$ onto a one-dimensional grid $i \in \mathcal{G} := \{0, \ldots, n_g - 1\}$ as

$$\rho_\sigma(i) = \underline{r}_\sigma + \frac{\bar{r}_\sigma - \underline{r}_\sigma}{n_g - 1} i \quad \text{and} \quad \gamma_\sigma(i) = l_\sigma - \rho_\sigma(i) \quad \forall \sigma \in \Sigma. \tag{25}$$

With the spectral radius $\mathcal{R}(A_\sigma)$, i.e., A_σ's largest absolute eigenvalue, the hyperparameters for our heuristic are given by Σ_s, $\mathcal{R}(A_\sigma) \leq l_\sigma, 0 \leq \underline{r}_\sigma \leq \bar{r}_\sigma \leq l_\sigma$. Again without proof, we conceptualize their meaning: l_σ sets the maximum admissible decay rate. Bounding it from below makes sense, as a sound abstraction must not decay faster than the underlying system. Imposing $\underline{r}_\sigma \leq \rho_\sigma(i)$ avoids high observer gains which amplify noise, i.e., they increase $\gamma_\sigma(i)$ and $\delta_\sigma(i)$. Lastly, $\rho_\sigma(i) \leq \bar{r}_\sigma$ sets the desired decay rate in the optimistic case.

The parameterization works as follows: for every grid point $i \in \mathcal{G}$, solve the SDP $\max v^*_{min}(i)$ in $P(i), W_\sigma(i), \beta^2_\sigma(i), \delta^2_\sigma(i)$, $v^*_\sigma(i)$ and $v^*_{min}(i)$ subject to (18)–(24) with the parameters set by (25), then compute $L_\sigma(i) = P^{-1}(i) W_\sigma(i)$. For all feasible SDPs, determine the actual abstraction coefficients (10) as the optimization variables do not reflect their actual values. Of all safe abstraction parametrizations, choose the one which maximizes the actual $v^*_{min} := \min_{\sigma \in \Sigma} v^*_\sigma$ obtained from (10).

4 Evaluation

In this section, we evaluate how introducing measurements lessens the blind abstractions' inherent pessimism and that the run-time policy (17) adapts well in a weakly-hard setting. Further, we demonstrate that mere stability is insufficient for guaranteeing safety and that correlated disturbances justify using a set-valued disturbance model. While this was carried out in a custom simulator, we complement the results with execution time measurements, which compare the abstractions' run-time overheads to those of set-valued estimators from the literature on an ARM Cortex-M4F processor.

4.1 Benchmark System

We use a double integrator as our benchmark system. The two modes are defined as *control* ($\sigma = 0$) and *skip* ($\sigma = 1$), i.e., measure both states and apply a state-space controller as the control signal or skip measurements and zero the controller output. They represent extreme examples for high and low criticality. We do not consider that measurement uncertainties enter the system via the controller and instead specify a single process disturbance affecting the states, which is the same mathematically. Further, we subject the measurements passed to the state abstraction to uncertainties. When executed, the controller places both closed-loop eigenvalues at 0.9. The safety specification is set arbitrarily to constrain both states and the control signal to a sphere of radius 1.25. The initial state is chosen as one tenth of this radius. For the lack of an analytical bound, we empirically chose the disturbances as high as possible such that our heuristic is still able to verify that $\sigma = 0$ is a safe mode, i.e., $v_{max} \leq v_0^*$. The benchmark's system model reads

$$\begin{cases} A_0 = \begin{bmatrix} 0.9950 & 8.205 \cdot 10^{-2} \\ -0.1100 & 0.8050 \end{bmatrix}, A_1 = \begin{bmatrix} 1 & 9.091 \cdot 10^{-2} \\ 0 & 1 \end{bmatrix} \\ G_0 = G_1 = \begin{bmatrix} -5.051 \cdot 10^{-3} \\ -0.1111 \end{bmatrix}, D_0 = D_1 = 3.516 \cdot 10^{-4} \\ C_0 = H_0 = I, Z_0 = 2.384 \cdot 10^{-5} \cdot I, \quad C_1, H_1, Z_1 \text{ empty} \\ X_0 = 1.563 \cdot 10^{-2} \cdot I, C_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.99 & 1.755 \end{bmatrix}, S = 1.563 \cdot I. \end{cases} \tag{26}$$

We draw the initial states independently and identically (i.i.d.) uniformly distributed from the surface of $\mathcal{E}(X_0)$.

4.2 Disturbance Scenarios

According to their elliptical bounds, we specify two types of disturbance scenarios for the simulation runs: in the *benign* case, we draw both disturbances uniformly and i.i.d. from their ellipsoids' volumes resulting in uncorrelated noise. In contrast, the *malicious* scenario aims to approximate the worst-case disturbance. We achieve this by drawing 100 points uniformly and i.i.d. from the corresponding ellipsoids' surfaces in each timestep. For the process disturbance, we then select the point d_k which maximizes $||C_s(x_k + G_{\sigma_k} d_k)||_S$ given the current state x_k. As the measurement uncertainty only affects the abstraction, we select z_k to maximize $||x_k + L_{\sigma_k} H_{\sigma_k} z_k||_P$. This procedure yields time-correlated values.

4.3 Effect of Measurements

In a first simulative experiment, we want to assess the effect of adding measurements as described in Section 3.3 by parametrizing three different state abstractions: the first one performs a *full measurement*, i.e., it utilizes both measurements via C_0 and H_0. The second abstraction uses *partial measurements* only, i.e., it disregards the measurement y_2 by omitting the second row of C_0 and H_0, respectively. Finally, the third abstraction is blind and thus disregards both y_1 and y_2, allowing it only to use the worst-case assumptions. Note that no measurements are used when skipping the controller.

Parametrizing each abstraction using our heuristic on an Intel Core i7-8565U took less than five seconds. Both observer abstractions were parametrized for $\underline{r}_0 = 0.3$, $\bar{r}_0 = 0.9$, $\lambda_0 = 0.94$, $\underline{r}_1 = 1.05$ and $\bar{r}_1 = \lambda_1 = 1.15$ on $n_g = 101$ grid points. For the blind abstraction, we

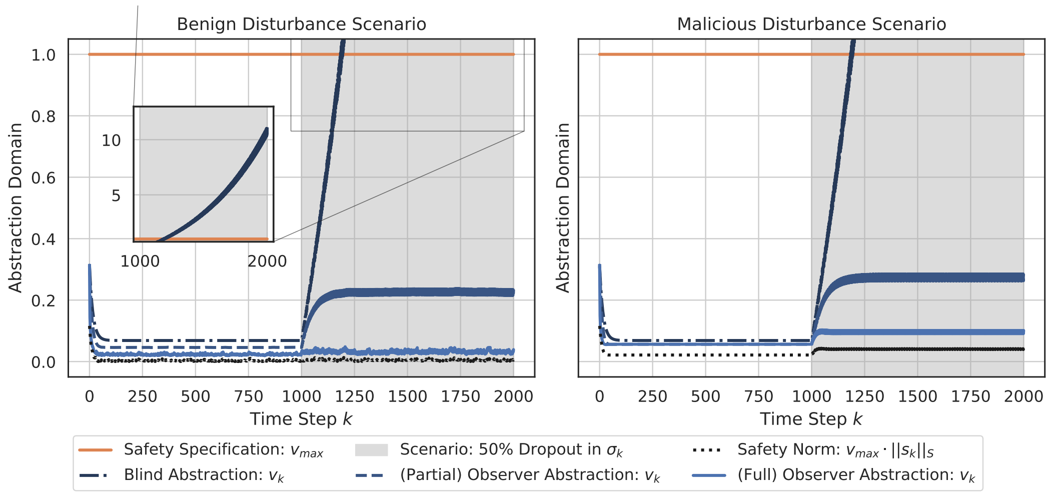

Figure 4 Effect of measurements. A double integrator is subjected to the benign (left) and malicious (right) disturbance scenario. Its state (dotted black) is compared to three state abstractions (blue), either running blindly (dash-dotted), measuring its partial (dashed), or full state (solid). The controller is always executed until time step 1000 and then subjected to 50% dropout, i.e., the modes are alternated deterministically.

Table 1 State abstraction coefficients for the double integrator and different measurement outputs computed by (10) and rounded to three significant digits.

Measurement	α	v_{max}	σ	ρ_σ	γ_σ	β_σ	δ_σ	v^*_σ
Full	0.314	1.00	0	0.396	0.554	0.0104	0.00616	1.05
			1	1.07	0.00	0.00425	0.00	0.934
Partial	0.317	1.00	0	0.900	0.0400	0.00448	0.000235	1.06
			1	1.07	0.00	0.00425	0.00	0.933
Blind	0.310	1.00	0	0.940	0.00	0.00411	0.00	1.06
			1	1.07	0.00	0.00411	0.00	0.933

matched $\bar{r}_0 = \lambda_0 = 0.94$ to make the SDP feasible. Table 1 shows the resulting abstraction coefficients computed via (10). The observer gains read $L_0 \approx \begin{bmatrix} 0.577 & 0.0487 \\ -0.0687 & 0.458 \end{bmatrix}$ for the full and $L_0 \approx \begin{bmatrix} 0.0393 \\ -0.0233 \end{bmatrix}$ for the partial measurement. The condition numbers of the shape matrices P^{-1} range between ~6.64 and ~7.48 with ~0.155 being the smallest of all their eigenvalues, i.e., the analysis ellipsoids are well-defined.

Figure 4 depicts two simulations in which the controller always runs at first and is then subjected to a deterministic 50% dropout sequence, i.e., the switching signal alternates between both modes starting at time step 1000. The benign or malicious disturbances perturb the system during the whole simulation, respectively. After the initial values have decayed, the blind abstraction's state is strictly higher than those of both observer abstractions. In the malicious scenario, the latter are nearly indistinguishable until the dropouts occur. There, the full measurement improves its abstraction noticeably. As we show later, the additional run-time overhead is negligible if the measurement is already available. Quantifying the overapproximation over 100 simulation runs by comparing the mean quotient of the individual v_k and $v_{max} \cdot ||s_k||_S$ yields factors between ~2.37 (full measurement, malicious disturbance,

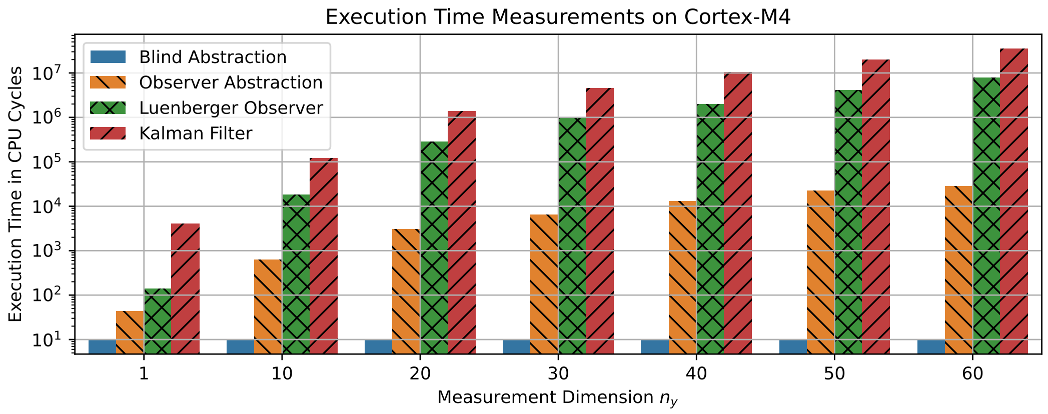

Figure 5 Execution times for updating a blind and observer abstraction by (6a) and (8a) as well as the update steps for a set-valued Luenberger observer [16] and Kalman filter [30] for different measurement sizes n_y. Times are given in CPU cycles on an ARM Cortex-M4F processor and plotted in logarithmic scales.

during dropout) and ~71.6 (partial measurement, benign disturbance, during dropout). This comparison leaves out the blind abstraction during dropout as – opposed to the observer abstractions – it appears to diverge. The large discrepancy signifies the benefit of separating the average case from the worst case at run time by reusing the controller's measurements. While our analysis cannot give guarantees about the steady states in case of dropouts, the simulations indicate that the observer abstractions average out well below their safety bound. The run-time algorithm (8) can predict upcoming violations in any case, such as for the blind abstraction around timestep 1200, allowing the scheduler to timely reconfigure the system for higher criticality.

4.4 Run-Time Overhead

While Section 4.3 focuses on the benefits of adding measurements, this section addresses their run-time overhead. For this, we measured the execution times of updating a blind and an observer abstraction by (6a) and (8a) given different measurement sizes $n_y \in \{1, \dots, 60\}$. To contrast the results, we also measured the cost of updating a set-valued Luenberger observer [16] and Kalman filter [30]. Most of the time, only a subset of the states is measured, i.e., $n_y < n_x$. We set $n_x = n_y$ for both observers, effectively reducing their execution time to a minimum. This does not affect the observer abstractions as the measurement term is independent of n_x, i.e., $||L_{\sigma_k} y_k||_P = \sqrt{y_k^T L_{\sigma_k}^T P L_{\sigma_k} y_k}$. We further reduced the cost for the Kalman filter by omitting the proposed line search for ω in [30, (13c)] in favor of a constant value while choosing ω by the trace criterion given in [16, (29)] for [30, (11c)].

Note that an additional online reachability analysis is required in all cases. While for state abstractions this reduces to trivial real-valued comparisions as outlined in Section 3.4, both set-valued observers from the literature require elaborate techniques as their ellipsoids' shapes are time-varying. As the references do not provide the respective analyses, we neither investigated this cost here nor their pessimism in Section 4.3. In all cases, we do not consider the acquisition of measurements as we assume that a state abstraction reuses the ones passed to the controller.

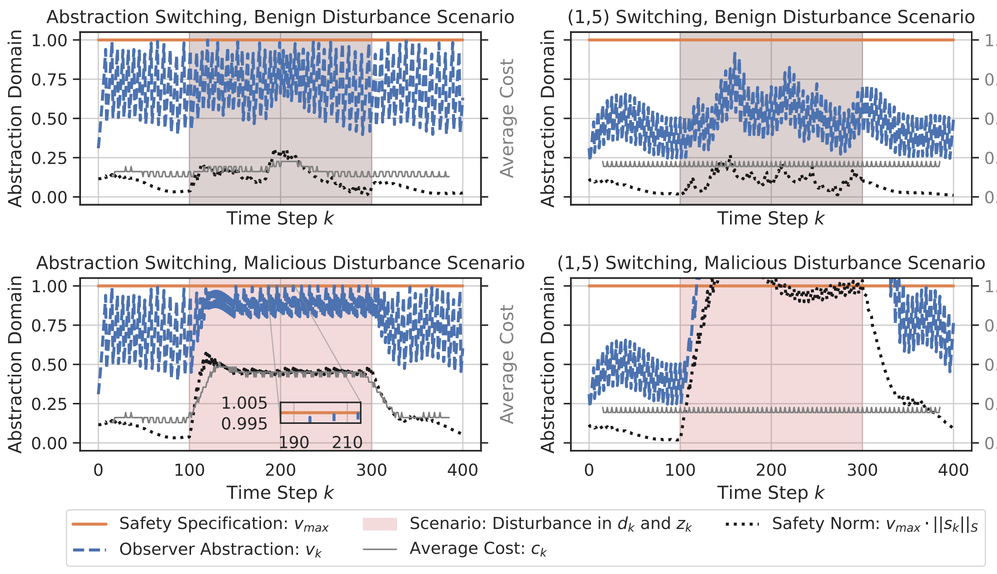

Figure 6 Comparison between the abstraction switching policy (17) and a minimal (1, 5) sequence (columns) subject to both disturbance scenarios (rows) when applied to the double integrator (26).

We conducted the experiment on an STM32F411E-Discovery board [37] with an ARM Cortex-M4F processor using `gcc` with `-O3 -ffp-contract=off`. Utilizing the DWT cycle counter, the measurement overhead amounted to one cycle consistently. We drew all coefficient values at random and computed the corresponding update steps in single precision for ten iterations utilizing some optimizations (e.g., take into account positive definiteness). We repeated this procedure for ten sets of coefficients, yielding 100 measurements per update step and n_y. Note that the parameters are stored in RAM. Placing them in flash adds additional wait states [37, p. 44]. This, however, affects all candidates. Figure 5 depicts the measurement results.

While the blind abstraction used 10 cycles consistently, the execution times for the other update steps rise for increasing measurement sizes. Note that this rise is not always monotonic as of compiler optimizations. The cycle counts for each size n_y coincide for all 100 execution time measurements of every individual update step (except for the Kalman filter for which the variations were negligible), which is to be expected as all are constant-time algorithms. The observer abstraction's maximum of 29874 cycles was attained at $n_y = 58$. In comparison, the Luenberger observer and Kalman filter took ~198 and ~1057 times longer for an update of this size. Considering the 96 MHz clock, updating the observer abstraction took around $300\mu s$ for a system with $n_y = 58$ measurement signals. Looking at the data, we conclude that the additional overhead from adding measurements is still negligible compared to the complexity imposed by a control system of this size.

4.5 Abstraction-Based Run-Time Switching and Insufficiency of Stability

Sections 4.3 and 4.4 evaluate the abstraction itself, i.e., how well it can track the physical state and how expensive the assessment is at run time. This is possible for any known switching sequence. In contrast, here we use the state abstraction as an inexpensive way to construct the switching sequence such that the system stays safe at all times by applying our prototypical policy (17). Further, we exemplify that stability alone is an insufficient criterion for safety when dealing with disturbances. These experiments were again carried out in simulation.

We only consider an abstraction utilizing both measurements. Without the more pessimistic configurations detailed in Section 4.3, we were able to increase the process uncertainty by a factor of 10 (i.e., $100 \cdot D_\sigma$) while keeping the heuristic working. To achieve this, we decreased $\bar{r}_0$ to 0.7, which was previously disregarded to keep the full and partial measurements comparable.

As the system is feasible and $\sigma = 0$ is safe, the policy guarantees safety at all time steps by Corollary 8, which was validated over 100 simulation runs. For the experiment, we compare it to a minimal $(1, 5)$-switching sequence, i.e., execute the controller only every fifth time. We decided on this specific sequence as its execution ratio is close to but higher than the average generated by the abstraction switching in the benign case. As shown by Theorem 9, the system is exponentially stable even under the more general $(1, 5)$ weakly-hard execution, i.e., skip no more than four consecutive controller executions. Even though our approach cannot guarantee safety under this sequence, the abstraction again remains a sound upper bound on the specification output as of Theorem 3. Subjecting each switching policy to both disturbance scenarios yields four combinations. Figure 6 depicts one of the 100 simulation runs obtained this way. As a visual guide for the switching sequence, the plots are overlaid with the average cost from controller execution defined by $c_k := 1 - \text{avg}(\sigma_k)$ where $\text{avg}(\cdot)$ is the moving average over 31 time steps.

While the abstraction stays below v_{max} for $(1, 5)$-switching under the benign disturbance, even the system itself violates its specification in the malicious case for all 100 runs indicating a critical condition. This shows that mere stability is insufficient in the presence of disturbances and that correlated disturbances require more pessimistic approaches, such as ellipsoidal models.

By design, the switching policy (17) keeps the abstraction and, by extension, the system within safe bounds at all time steps. While the average cost stays nearly constant at 15% to 20% of the controller executions during the benign disturbance, it adapts to the malicious case by increasing the average cost to around 45% after a brief period of overshoot between timesteps 100 and 150. From this, we conclude that our state abstractions are a pessimistic yet cheap and simple enough tool for monitoring and influencing run-time adaptive switched linear systems at the operating system level.

5 Related Work

Adaptive scheduling and selective verification of CPSs is a thriving field of research. Our approach shares its objective correspondingly with a wide area of related work that we inherently expand upon as well. Yet, we are not aware of any approach that provides (1) sound yet very cost-effective state abstraction, (2) guarantees not only stability but also adhering to safety constraints, and (3) uses observer-based feedback of measured values to mitigate pessimism. The following papers each differ from our approach in one or more respects.

We are not the first to note that CPSs are a combination of criticalities in time and quality of control (QoC), thus sharing parallels with MCS. For example, in [34], the QoC is maximized while preserving guarantees for cyber tasks, while [26] maps high-level hazards to the criticality of tasks.

The co-design of the controller and real-time system was introduced by Seto et al. [36] to improve utilization by relaxing timing requirements. Realizing the impact on QoC [4], [31] proposed to adjust control parameters to counteract deadline misses. For example, by adapting the control period and deadline adherence [10, 11, 17, 20]. A popular variant of mitigated timing constraints is (m, K)-firm scheduling introduced by [21]. Subsequently, more flexible task models and switching between safe and optimistic control were researched [15, 47, 43, 5, 35]. Likewise, for sporadic bursts of deadline misses, control parameters can be adapted [32, 14] and [44, 29] provide stability analysis of closed-loop systems.

Conversely, optimal sampling period approaches [8, 12] infer a sampling period to maximize QoC and minimize the quadratic cost function regardless of other tasks.

Even more radical are approaches on self-triggered control [3, 39, 22, 23], which analyze the system state to predict the next control instant. These approaches offer superior average-case performance – unfortunately – to the detriment of overall schedulability.

The field of model predictive control emerged from the requirement to impose bounds on control systems. Naturally, our approach reuses its fundamental concepts, e.g. bounding the reachable sets of a perturbed system by ellipsoids as in Tube Model Predictive Control [9]. In fact, it can be interpreted as a stripped down version of MPC. While utilizing information about the full system state promises to be less pessimistic and there are even variants available which incorporate scheduling decisions [24, 25, 28], they require solving potentially large optimization problems at run time. Further, determining a discrete switching sequence can yield a problem growing exponentially in the prediction horizon [1]. This poses a high burden on their implementation in embedded control systems [27]. We argue that while superior in performance, MPC is only viable if the control system requires it anyway.

6 Conclusion

The physical state, or rather its distance from the safety specification, is a crucial criterion for the actual criticality of control applications in many real-world applications. In this paper, we proved that careful abstraction allows for state estimation efficiently enough to serve as run-time monitoring. Therefore, we extended our previously defined one-dimensional convergence rate abstractions described in [19] to linear systems which feature multiple modes of controller execution and are subject to ellipsoidally bound disturbances. Further, we introduced the concept of observer abstractions that allow for the feedback of uncertain measurements to lessen overapproximation, making the abstractions much more practical. Given a well-posed specification, we then developed a design-time analysis for their parameterization. Finally, as part of a case study, we validated our disturbance model, showed that the benefits of introducing measurements outweighs their run-time overhead, and illustrated that our prototypical run-time policy for optimistically choosing the mode of controller execution adapts well to changing disturbances while obeying safety specifications even in the worst case. We consider our work a foundation for designing mixed-criticality systems and scheduling approaches that leverage a system's physical state for control tasks as an additional criticality criterion.

References

1 Ricardo P. Aguilera and Daniel E. Quevedo. On the stability of MPC with a Finite Input Alphabet. *IFAC Proceedings Volumes*, 44(1):7975–7980, 2011. `doi:10.3182/20110828-6-IT-1002.02705`.

2 Anayo K. Akametalu, Claire J. Tomlin, and Mo Chen. Reachability-Based Forced Landing System. *Journal of Guidance, Control, and Dynamics*, 41(12):2529–2542, 2018. `doi:10/gfpcbn`.

3 Shigeru Akashi, Hideaki Ishii, and Ahmet Cetinkaya. Self-triggered control with tradeoffs in communication and computation. *Automatica*, 94:373–380, 2018. `doi:10.1016/j.automatica.2018.04.028`.

4 K.-E. Årzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 5, pages 4865–4870, 2000. `doi:10/ck4mjj`.

5 Stanley Bak, Deepti K. Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo, and Lui Sha. The system-level simplex architecture for improved real-time embedded system safety. In *Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 99–107, 2009. `doi:10.1109/RTAS.2009.20`.

6 André Benine-Neto, Stefano Scalzi, Saïd Mammar, and Mariana Netto. Dynamic controller for lane keeping and obstacle avoidance assistance system. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1363–1368, 2010. `doi:10/bj4xpb`.

7 Guillem Bernat, Alan Burns, and Alberto Liamosi. Weakly Hard Real-time Systems. *IEEE Transactions on Computers*, 50(4):308–321, 2001. `doi:10/cqd6d3`.

8 Enrico Bini and Giuseppe M. Buttazzo. The optimal sampling pattern for linear control systems. *IEEE Transactions on Automatic Control*, 59(1):78–90, 2014. `doi:10/f3nxws`.

9 Mark Cannon, Johannes Buerger, Basil Kouvaritakis, and Saša Rakovic. Robust Tubes in Nonlinear Model Predictive Control. *IEEE Transactions on Automatic Control*, 56(8):1942–1947, 2011. `doi:10/dhg386`.

10 Rosa Castañé, Pau Marti, Manel Velasco, Anton Cervin, and Daniel Henriksson. Resource management for control tasks based on the transient dynamics of closed-loop systems. In *Proceedings of the 18th Euromicro Conf. on Real-Time Systems (ECRTS '06)*, pages 172–182, Los Alamitos, CA, USA, 2006. `doi:10/bx9rps`.

11 Anton Cervin, Johan Eker, Bo Bernhardsson, and Karl-Erik Årzén. Feedback–feedforward scheduling of control tasks. *Real-Time Systems*, 23(1-2):25–53, 2002. `doi:10/fnb5nk`.

12 Anton Cervin, Manel Velasco, Pau Martí, and Antonio Camacho. Optimal online sampling period assignment: Theory and experiments. *IEEE Trans. on Control Systems Technology*, 19(4):902–910, 2011. `doi:10/d38qtf`.

13 Long Cheng, Kai Huang, Gang Chen, Biao Hu, and Alois Knoll. Mixed-criticality control system with performance and robustness guarantees. In *Proceedings of the IEEE Trustcom/BigDataSE/ICESS*, pages 767–775, 2017. `doi:10/gr65zt`.

14 Hoon Sung Chwa, Kang G. Shin, and Jinkyu Lee. Closing the gap between stability and schedulability: A new task model for cyber-physical systems. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 327–337, 2018. `doi:10/gqz7sd`.

15 Xiaotian Dai, Wanli Chang, Shuai Zhao, and Alan Burns. A Dual-Mode Strategy for Performance-Maximisation and Resource-Efficient CPS Design. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):85:1–85:20, October 2019. `doi:10/gq5wfw`.

16 Monia Dkhil, Thach Ngoc Dinh, Zhenhua Wang, Tarek Raïssi, and Messaoud Amairi. Interval Estimation for Discrete-Time Switched Linear Systems Based on L_∞ Observer and Ellipsoid Analysis. *IEEE Control Systems Letters*, 5(1):13–18, 2021. `doi:10/gq5wdm`.

17 Daniele Fontanelli, Luca Greco, and Luigi Palopoli. Soft real-time scheduling for embedded control systems. *Automatica*, 49(8):2330–2338, 2013. `doi:10/gq5wfr`.

18 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 (HSCC '18)*, pages 91–100, New York, NY, USA, April 2018. ACM. `doi:10/gq5wdx`.

19 Maximilian Gaukler, Tim Rheinfels, Peter Ulbrich, and Günter Roppenecker. Convergence Rate Abstractions for Weakly-Hard Real-Time Control, 2019. `doi:10/gq5wdt`.

20 Luca Greco, Daniele Fontanelli, and Antonio Bicchi. Design and Stability Analysis for Anytime Control via Stochastic Scheduling. *IEEE Transactions on Automatic Control*, 56(3):571–585, March 2011. `doi:10/bgwcfj`.

21 Moncef Hamdaoui and Parameswaran Ramanathan. A Dynamic Priority Assignment Technique for Streams with (m, K)-firm Deadlines. *IEEE Transactions on Computers*, 44(12):1443–1451, 1995. doi:10/cq995j.

22 W. P. M. H. Heemels and M. C. F. Donkers. Model-based periodic event-triggered control for linear systems. *Automatica*, 49(3):698–711, 2013. doi:10/f4sp65.

23 W. P. M. H. Heemels, Karl Henrik Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3270–3285, 2012. doi:10/f22pf7.

24 Dan Henriksson, Anton Cervin, Johan Åkesson, and Karl-Erik Årzén. Feedback scheduling of model predictive controllers. In *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 207–216, 2002. doi:10/frxj6d.

25 Erik Henriksson, Daniel E. Quevedo, Henrik Sandberg, and Karl Henrik Johansson. Self-Triggered Model Predictive Control for Network Scheduling and Control. *IFAC Proceedings Volumes*, 45(15):432–438, 2012. doi:10/f24cvt.

26 Viacheslav Izosimov and Erik Levholt. Mixed criticality metric for safety-critical cyber-physical systems on multi-core architectures. *Methods*, 2:8, 2015.

27 Tor Arne Johansen. Toward dependable embedded model predictive control. *IEEE Systems Journal*, 11(2):1208–1219, 2017. doi:10.1109/JSYST.2014.2368129.

28 Yingzhao Lian, Yuning Jiang, Naomi Stricker, Lothar Thiele, and Colin N. Jones. Robust resource-aware self-triggered model predictive control. *IEEE Control Systems Letters*, 6:1724–1729, 2022. doi:10/gr65zv.

29 Martina Maggio, Arne Hamann, Eckart Mayer-John, and Dirk Ziegenbein. Control-System Stability Under Consecutive Deadline Misses Constraints. In Marcus Völp, editor, *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, volume 165 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:24, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10/ghqcvc.

30 Benjamin Noack, Marcus Baum, and Uwe D. Hanebeck. State estimation for ellipsoidally constrained dynamic systems with set-membership pseudo measurements. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 297–302, 2015. doi:10/gq5wfg.

31 Paolo Pazzaglia, Arne Hamann, Dirk Ziegenbein, and Martina Maggio. Adaptive design of real-time control systems subject to sporadic overruns. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1887–1892, 2021. doi:10/gq5whh.

32 Paolo Pazzaglia, Claudio Mandrioli, Martina Maggio, and Anton Cervin. DMAC: Deadline-Miss-Aware Control. In Sophie Quinton, editor, *Proceedings of the 31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, volume 133 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:24, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10/gf6j4p.

33 Gilberto Pin, Peng Li, Giuseppe Fedele, and Thomas Parisini. A deadbeat observer for LTI systems by time/output-dependent state mapping. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4795–4800, 2017. doi:10.1109/CDC.2017.8264367.

34 Reinhard Schneider, Dip Goswami, Alejandro Masrur, Martin Becker, and Samarjit Chakraborty. Multi-layered Scheduling of Mixed-criticality Cyber-physical Systems. *Journal of System Architecture*, 59(10):1215–1230, November 2013. doi:10/f5qdjc.

35 Danbing Seto, Bruce H. Krogh, Lui Sha, and Alongkarn Chutinan. Dynamic control system upgrade using the simplex architecture. *IEEE Control Systems*, 18(4):72–80, August 1998. doi:10/fk87g6.

36 Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. On Task Schedulability in Real-time Control Systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS '96)*, pages 13–21, Los Alamitos, CA, USA, December 1996. doi:10.1109/REAL.1996.563693.

37 STMicroelectronics. *Reference manual RM0383, Rev 3*, 2018.

38 Mohammad M. Sultan, Daniel Biediger, Bernard Li, and Aaron T. Becker. The Reachable Set of a Drone: Exploring the Position Isochrones for a Quadcopter. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7679–7685, 2021. `doi:10.1109/ICRA48506.2021.9561715`.

39 Sebastian Trimpe and Raffaello D'Andrea. Event-based state estimation with variance-based triggering. *IEEE Transactions on Automatic Control*, 59(12):3266–3281, 2014. `doi:10.1109/TAC.2014.2351951`.

40 Sezai Emre Tuna. Deadbeat Observer: Construction via Sets. *IEEE Transactions on Automatic Control*, 57(9):2333–2337, 2012. `doi:10.1109/TAC.2012.2183197`.

41 Lieven Vandenberghe and Stephen Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, 1996. `doi:10.1137/1038003`.

42 Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS '07)*, pages 239–243, 2007. `doi:10/cg89kh`.

43 Prasanth Vivekanandan, Gonzalo Andres Garcia, Heechul Yun, and Shawn Shahriar Keshmiri. A simplex architecture for intelligent and safe unmanned aerial vehicles. In *Proceedings of the IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 69–75, 2016. `doi:10.1109/RTCSA.2016.17`.

44 Nils Vreman, Anton Cervin, and Martina Maggio. Stability and Performance Analysis of Control Systems Subject to Bursts of Deadline Misses. In Björn B. Brandenburg, editor, *Proceedings of the 33rd Euromicro Conference on Real-Time Systems*, volume 196 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:23, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ECRTS.2021.15`.

45 Nils Vreman, Paolo Pazzaglia, Victor Magron, Jie Wang, and Martina Maggio. Stability of Linear Systems Under Extended Weakly-Hard Constraints. *IEEE Control Systems Letters*, 6:2900–2905, 2022. `doi:10/gqnxtm`.

46 Guorong Wang, Yimin Wei, and Sanzheng Qiao. *Generalized Inverses: Theory and Computations*. Springer Singapore, first edition, 2018. `doi:10/gq5wd4`.

47 Xiaofeng Wang, Naira Hovakimyan, and Lui Sha. Rsimplex: A robust control architecture for cyber and physical failures. *ACM Transactions on Cyber-Physical Systems*, 2(4), July 2018. `doi:10.1145/3121428`.

A Proofs

While in Section 3 we focused on conveying the ideas and design decisions behind our approach, here we formalize and prove the state abstractions' properties.

▶ **Lemma 1** (Generalization). *By setting* $L_\sigma = 0 \quad \forall \sigma \in \Sigma$, *the observer abstraction* (8) *is a true generalization of the blind abstraction* (6).

Proof. When applying norm homogeneity (3a), the observer abstraction's dynamics (8a) and their coefficients (10) are reduced to the blind counterparts (6a) and (7) by setting $L_\sigma = 0 \quad \forall \sigma \in \Sigma$. ◀

▶ **Lemma 2** (Disturbance Bound). *The highest impact of an ellipsoidally constrained disturbance in terms of the P-norm is attained at the ellipsoid's surface:*

$$\max_{||x||_Q \le 1} ||Mx||_P = \max_{||x||_Q = 1} ||Mx||_P \overset{(1c)}{=} ||M||_{PQ}. \tag{27}$$

Proof. We prove destructively: assume some $\tilde{x}$ with $||\tilde{x}||_Q < 1$ were a maximizer on the ellipsoid's inside. Then $x := ||\tilde{x}||_Q^{-1}\tilde{x}$ is of unit length (i.e., it lies on the surface of Q) and by the norm properties

$$||Mx||_P \overset{(3a)}{=} \underbrace{||\tilde{x}||_Q^{-1}}_{>1} ||M\tilde{x}||_P \overset{(3b)}{>} ||M\tilde{x}||_P$$

is greater than the assumed maximizer. This contradiction concludes (27). ◀

▶ **Theorem 3** (Soundness). *For any switching sequence $\sigma_k \in \Sigma, k \in \mathbb{N}_0$, an (observer) abstraction* (8) *poses a sound upper bound for the underlying system's state* (4)*, i.e.,*

$$||x_k||_P \leq v_k \quad \forall k \in \mathbb{N}_0. \tag{28}$$

Proof. We prove (28) by induction.

Base Case. By the ellipsoidal constraint on the initial state, the norm properties, and the definition of blind abstractions

$$||x_0||_P \overset{(4d)}{\leq} \max_{||x||_{X_0} \leq 1} ||x||_P \overset{(27)}{=} \underbrace{||I||_{PX_0}}_{\overset{(10)}{=} \alpha} \overset{(8b)}{=} v_0.$$

Inductive Assumption (I.A.). $||x_k||_P \leq v_k$.

Inductive Step ($k \to k+1$). We incorporate the observer by adding a zero and substituting the measurement. Applying the triangle inequality and bounding the summands by the norm properties yields the abstraction's coefficients and dynamics by definition:

$$||x_{k+1}||_P \overset{(4a)}{=} ||A_{\sigma_k}x_k + G_{\sigma_k}d_k||_P \overset{(9),+0}{=} ||\tilde{A}_{\sigma_k}x_k + G_{\sigma_k}d_k + L_{\sigma_k}C_{\sigma_k}x_k||_P \overset{(4b)}{=}$$
$$||\tilde{A}_{\sigma_k}x_k + G_{\sigma_k}d_k - L_{\sigma_k}H_{\sigma_k}z_k + L_{\sigma_k}y_k||_P \overset{(3c)}{\leq} ||\tilde{A}_{\sigma_k}x_k||_P + ||G_{\sigma_k}d_k||_P + ||L_{\sigma_k}H_{\sigma_k}z_k||_P$$
$$+ ||L_{\sigma_k}y_k||_P \overset{(1c),(4c),(27)}{\leq} \underbrace{||\tilde{A}_{\sigma_k}||_{PP}}_{\overset{(10)}{=} \rho_{\sigma_k}} ||x_k||_P + \underbrace{||G_{\sigma_k}||_{PD_{\sigma_k}} + ||L_{\sigma_k}H_{\sigma_k}||_{PZ_{\sigma_k}}}_{\overset{(10)}{=} \beta_{\sigma_k}} + ||L_{\sigma_k}y_k||_P \overset{I.A.}{\leq}$$
$$\rho_{\sigma_k}v_k + \beta_{\sigma_k} + ||L_{\sigma_k}y_k||_P \overset{(8a)}{=} v_{k+1}.$$
◀

▶ **Theorem 4** (Safety Bound). *If an abstraction's state is within its safety bound, the system is guaranteed to obey its specification, i.e.,*

$$v_k \leq v_{max} \Rightarrow ||s_k||_S \leq 1 \quad \forall k \in \mathbb{N}_0. \tag{29}$$

Proof. After expanding the definitions, we apply Theorem 3 and the norm consistency. Assume $v_k \leq v_{max} (*)$, then

$$||s_k||_S \overset{(4e)}{=} ||C_s x_k||_S \overset{(1c)}{\leq} \underbrace{||C_s||_{SP}}_{\overset{(10)}{=} v_{max}^{-1}} ||x_k||_P \overset{(28)}{\leq} v_{max}^{-1} v_k \overset{(*)}{\leq} 1.$$
◀

▶ **Lemma 5** (Worst-case Behavior). *The dynamic system* (11) *provides an upper bound on the observer abstraction* (8), *i.e.,*

$$v_k \leq \bar{v}_k \quad \forall k \in \mathbb{N}_0. \tag{30}$$

Proof. First, we decompose the measurement term and apply Theorem 3:

$$||L_{\sigma_k} y_k||_P \overset{(4b)}{\leq} ||L_{\sigma_k} C_{\sigma_k} x_k + L_{\sigma_k} H_{\sigma_k} z_k||_P \overset{(1c),(3c),(4c),(27)}{\leq}$$
$$\underbrace{||L_{\sigma_k} C_{\sigma_k}||_{PP}}_{\overset{(10)}{=}\gamma_{\sigma_k}} ||x_k||_P + \underbrace{||L_{\sigma_k} H_{\sigma_k}||_{PZ_{\sigma_k}}}_{\overset{(10)}{=}\delta_{\sigma_k}} \overset{(28)}{\leq} \gamma_{\sigma_k} v_k + \delta_{\sigma_k}. \tag{31}$$

We then prove (30) by induction:
Base Case: By definition, $v_0 \overset{(8b),(11b)}{=} \bar{v}_0$.
Inductive Assumption (I.A.): $v_k \leq \bar{v}_k$.
Inductive Step ($k \to k+1$):

$$v_{k+1} \overset{(8a),(31)}{\leq} (\rho_{\sigma_k} + \gamma_{\sigma_k}) v_k + \beta_{\sigma_k} + \delta_{\sigma_k} \overset{I.A.}{\leq} (\rho_{\sigma_k} + \gamma_{\sigma_k}) \bar{v}_k + \beta_{\sigma_k} + \delta_{\sigma_k} \overset{(11a)}{=} \bar{v}_{k+1}.$$ ◀

▶ **Lemma 6** (An Inclusion Condition). *The feasible set* (13) *obeys the following inclusion condition:*

$$a \leq b \Rightarrow \Sigma_f(b) \subseteq \Sigma_f(a). \tag{32}$$

Proof.

$$\Sigma_f(b) \overset{(13)}{=} \{\sigma \in \Sigma | b \leq v_\sigma^*\} \overset{a \leq b}{=} \{\sigma \in \Sigma | a \leq v_\sigma^* \vee b \leq v_\sigma^*\} \subseteq \{\sigma \in \Sigma | a \leq v_\sigma^*\} \overset{(13)}{=} \Sigma_f(a).$$ ◀

▶ **Theorem 7** (Recursive Feasibility). *Under the conditions* (14) *and* (15), *the switching policy* (16) *is recursivly feasible, i.e.,* $\Sigma_f(v_k) \neq \emptyset \; \forall k \in \mathbb{N}_0$. *Assume (w.l.o.g.) that* $v_0^* \geq v_{max}$. *Then,*

$$0 \in \Sigma_f(v_k) \neq \emptyset \quad \forall k \in \mathbb{N}_0. \tag{33}$$

Proof. In preparation, observe that

$$\sigma \in \Sigma_f(v_k) \overset{(13)}{\Leftrightarrow} v_k \leq v_\sigma^* \overset{(12)}{=} \frac{v_{max} - \beta_\sigma - \delta_\sigma}{\rho_\sigma + \gamma_\sigma} \Leftrightarrow (\rho_\sigma + \gamma_\sigma) v_k + \beta_\sigma + \delta_\sigma \leq v_{max}. \tag{34}$$

We prove (33) by induction using the results from Lemma 6:
Base Case: By assuming (14) and (15), $v_0 \leq v_{max} \leq v_0^*$. Therefore:

$$0 \overset{(13)}{\in} \Sigma_f(v_0^*) \overset{(32)}{\subseteq} \Sigma_f(v_0).$$

Inductive Assumption (I.A.): $0 \in \Sigma_f(v_k)$.
Inductive Step ($k \to k+1$): Assuming the I.A. holds, we know that $\Sigma_f(v_k) \neq \emptyset$, therefore, using any policy $\sigma_k \in \Sigma_f(v_k)$ leads to

$$v_{k+1} \overset{(8a),(31)}{\leq} (\rho_{\sigma_k} + \gamma_{\sigma_k}) v_k + \beta_{\sigma_k} + \delta_{\sigma_k} \leq \max_{\sigma \in \Sigma_f(v_k)} ((\rho_\sigma + \gamma_\sigma) v_k + \beta_\sigma + \delta_\sigma) \overset{(34)}{\leq} v_{max} \overset{(15)}{\leq} v_0^*. \tag{35}$$

Using the definition of feasible sets, we arrive at $v_{k+1} \leq v_0^* \overset{(13)}{\Leftrightarrow} 0 \in \Sigma_f(v_{k+1})$. ◀

▶ **Corollary 8** (Guaranteed Safety). *Under the conditions of Theorem 7, any switching policy obeying* (16) *guarantees the safety specification* (4f) $\forall k \in \mathbb{N}_0$.

Proof. $v_k \le v_{max}$ holds for $k = 0$ due to (14) and for $k \in \mathbb{N}$ due to (15) and (33)–(35). Theorem 4 concludes (4f) $\forall k \in \mathbb{N}_0$. ◀

▶ **Theorem 9** (Double Integrator Stability). *In the absence of disturbances ($d_k = 0, z_k = 0 \forall k \in \mathbb{N}_0$), the double integrator benchmark* (26) *is exponentially stable under* $(1, 5)$*-switching, i.e., there exist some $C > 0, 0 < \lambda < 1$ such that*

$$||x_k||_P \le C\lambda^k ||x_0||_P. \tag{36}$$

Proof. Given $d_k = 0, z_k = 0 \forall k \in \mathbb{N}_0$, the system dynamics (4a) have the explicit solution

$$||x_k||_P = ||\left(\prod_{i=0}^{k-1} A_{\sigma_i}\right) x_0||_P. \tag{37}$$

For any switching sequence satisfying the $(1, 5)$ constraint, we can decompose the product (37) into n shorter sequences of the shape $M_m = A_0 A_1^m$ and some open loop part A_1^l with $m, l \in \{0, \dots, 4\}$. Let $f(i)$ be the mapping for the concrete sequence $(**)$. Applying the consistency properties yields

$$||x_k||_P \overset{(37),(**)}{=} ||A_1^l \left(\prod_{i=1}^{n} M_{f(i)}\right) x_0||_P \overset{(1c),(1d)}{\le} ||A_1^l||_{PP} \left(\prod_{i=1}^{n} ||M_{f(i)}||_{PP}\right) ||x_0||_P$$

$$\overset{(3b),n\ge 0}{\le} \underbrace{\max_{l\in\{0,\dots,4\}} ||A_1^l||_{PP}}_{=:\tilde{C}} (\underbrace{\max_{m\in\{0,\dots,4\}} ||M_m||_{PP}}_{=:\tilde{\lambda}})^n ||x_0||_P.$$

Given that the M_m describe between one and five timesteps, we can bound n from below as $\lfloor \frac{k}{5} \rfloor \le n$. Assuming $0 \le \tilde{\lambda} \le 1$, the exponential decay can be upper bounded as

$$\tilde{C}\tilde{\lambda}^n \le \tilde{C}\tilde{\lambda}^{\lfloor \frac{k}{5} \rfloor} \overset{\lfloor \frac{k}{5} \rfloor \ge \frac{k}{5}-1}{=} \tilde{C}\tilde{\lambda}^{\frac{k}{5}-1} = \underbrace{\tilde{C}\tilde{\lambda}^{-1}}_{=:C}(\underbrace{\tilde{\lambda}^{\frac{1}{5}}}_{=:\lambda})^k.$$

Combining the above yields the stability condition (36), i.e., $||x_k||_P \le \tilde{C}\tilde{\lambda}^n ||x_0||_P \le C\lambda^k ||x_0||_P$.

The last step is to prove that $0 \le \tilde{\lambda} \le 1$ which can be shown for $P := \begin{bmatrix} 5.849 & 5.045 \\ 5.045 & 13.45 \end{bmatrix} > 0$ and (1b) yielding $\tilde{\lambda} = 0.9712$ and therefore stability with $C = 1.191$ and $\lambda = 0.9942$. ◀