


# Offline and online monitoring of scattered uncertain logs using uncertain linear dynamical systems<sup>\*</sup>

Bineet Ghosh<sup>1</sup>  and Étienne André<sup>2</sup> 

<sup>1</sup> The University of North Carolina at Chapel Hill, NC, The United States of America

<sup>2</sup> Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

**Abstract.** Monitoring the correctness of distributed cyber-physical systems is essential. We address the analysis of the log of a black-box cyber-physical system. Detecting possible safety violations can be hard when some samples are uncertain or missing. In this work, the log is made of values known with some uncertainty; in addition, we make use of an over-approximated yet expressive model, given by a non-linear extension of dynamical systems. Given an offline log, our approach is able to monitor the log against safety specifications with a limited number of false alarms. As a second contribution, we show that our approach can be used online to minimize the number of sample triggers, with the aim at energetic efficiency. We apply our approach to two benchmarks, an anesthesia model and an adaptive cruise controller.

**Keywords:** energy-aware monitoring, cyber-physical systems, formal methods

## 1 Introduction

The pervasiveness of distributed cyber-physical systems is highly increasing, accompanied by associated safety concerns. Formal verification techniques usually require a (white-box) model, which is not often available, because some components are black-box, or because the entire system has no formal model. In addition, formal verification techniques for cyber-physical systems are often subject to state space explosion, often preventing a satisfactory scalability. Therefore, *monitoring*, as a lightweight yet feasible verification technique, can bring practical results of high importance for larger models.

Monitoring aims at analyzing the log of a concrete system, so as to deduce whether a specification (e.g., a safety property) is violated. Monitoring can be done *offline* (i.e., after the system execution, assuming the knowledge of the entire log), or *online* (at runtime, assuming a partial log). When the log is an aperiodic timed sequence of valuations of continuous variables, with a logging

---

<sup>\*</sup> This work is partially supported by the ANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015), and the National Science Foundation (NSF) of the United States of America under grant number 2038960.

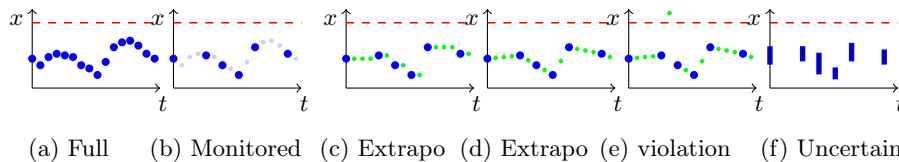


Fig. 1: Monitoring at discrete time steps

not occurring at every discrete time step, and when the system under monitoring is a black box, a major issue is: how to be certain that, in between two discrete valuations, the specification was not violated at another discrete time step at which no logging was performed? For example, consider a system for which a logging at every discrete time step would yield the log depicted in Fig. 1a. Assume the logging was done at only *some* time steps, given in Fig. 1b, due to some sensor faults, or to save energy with only a sparse, scattered logging. How to be certain that, in between two discrete samples, another discrete sample (not recorded) did not violate the specification? For example, by just looking at the discrete samples in Fig. 1b, there is no way to formally guarantee that the unsafe zone (i.e., above the red, dashed line) was never reached by another discrete sample which was not recorded. In many practical cases, a piecewise-constant or linear approximation (see, e.g., Figs. 1c and 1d, where the large blue dots denote actual samples, while the small green dots denote reconstructed samples using some extrapolation) is arbitrary and not appropriate; even worse, it can yield a “safe” answer, while the actual system could actually have been unsafe at some of the missing time steps. On the contrary, assuming a completely arbitrary dynamics will always yield “potentially unsafe”—thus removing the interest of monitoring. For example, from the samples in Fig. 1b, without any knowledge of the model, one can always envision the situation in Fig. 1e, which shows the variable  $x$  crossing the unsafe region (dashed) at some unlogged discrete time step—even though this is unlikely if the dynamics is known to vary “not very fast”.

*Contributions* In this work, we address the problem of performing monitoring over a set of scattered and *uncertain* samples. First, we cope with uncertainties from the sensors by allowing for *uncertain* samples, given by zonotopes over the continuous variables; that is, at each logged timestamp, the log gives not a constant value for the continuous variables, but a *zonotope*. A simple case of an uncertain log over a single variable  $x$  is depicted in Fig. 1f in the form of simple intervals. The timestamp at each discrete sample of the log is however supposed to be constant (i.e., a single point). Second, to over-approximate the system behavior, and in the spirit of the “model-bounded monitoring” proposed in [34], we use an extension of *linear dynamical systems*, extended with uncertainty, i.e., allowing *uncertainty* in the dynamics matrix [22]. Having some over-approximated knowledge of the system is a natural assumption in practice: when monitoring a car, one generally knows an upper-bound on its maximum speed, or on its max-

imum acceleration (perhaps depending on its current speed). To cope with the liberal dynamics of our extension of linear dynamical systems, we use a recent technique [18], that performs an efficient reachability analysis for such uncertain linear dynamical systems. The use of such an over-approximation of the actual system is the crux of our approach, allowing us to discard unlikely behaviors, such as the unlikely safety violation depicted in Fig. 1e.

Our first main contribution is to propose a new rigorous analysis technique for offline monitoring of safety properties over scattered *uncertain* samples, using uncertain linear systems as an over-approximation of the system. This over-approximation allows us to extrapolate the behavior since the latest known sample, and to rule out safety violations at some missing discrete samples. Note that our approach uses some discrete analysis as underlying reachability computation technique, and will not however guarantee the absence of safety violations at arbitrary (continuous) timestamps; its main advantage is to offer formal guarantees in the context of missing discrete samples for a given logging granularity.

Our second main contribution focuses on *energy-efficient online monitoring*. For each recorded sample, we run a reachability analysis, and we derive the smallest next discrete time step  $t$  in the future at which the safety property may be violated depending on the latest known sample and the over-approximated model dynamics. In a context in which monitoring simply observes the behavior and does not lead to corrective actions, any sample before  $t$  is useless because we *know* from the over-approximated model dynamics that no safety violation can happen before  $t$ . Therefore, we can schedule the next sample at time  $t$ , which reduces the number of discrete samples, and therefore the energy consumption and bandwidth use. We show that our method is correct, i.e., we can safely discard discrete samples without missing any unsafe behavior. We show the practical applicability of our approach on two benchmarks: an anesthesia model, and an adaptive cruise controller.

*Outline* We review related works in Section 2. We recall uncertain linear dynamical systems in Section 3. We introduce our (offline and online) monitoring frameworks in Section 4, and run experiments in Section 5.

## 2 Related works

*Monitoring* Monitoring complex systems, and notably cyber-physical systems, drew a lot of attention in the last decades, e.g., [23,6,5,34,24]. In parallel to monitoring specifications using signal temporal logics (see e.g., [13,20,29]), monitoring using automata-based specifications drew recent attention. Complex, quantitative extensions of automata were studied in the recent years: after timed pattern matching on timed regular expressions [31] was proposed by Ulus *et al.*, Waga *et al.* proposed a technique for timed pattern matching [32] (with an additional work by Bakhirkin *et al.* [4]) and then for parametric timed pattern matching [3,33,35], with application to offline monitoring.

In [34], we proposed *model-bounded monitoring*: instead of monitoring a black-box system against a sole specification, we use in addition a (limited, over-approximated) knowledge of the system, to eliminate false positives. This over-approximated knowledge is given in [34] in the form of a *linear hybrid automaton* (LHA) [19], an extension of finite-state automata with continuous variables; their flow in each location (“mode”) is given as a linear constraint over derivatives; location invariants and transition guards are given by linear constraints over the system variables. We use in [34] both an *ad-hoc* implementation, and another one based on PHAVerLite [7]. In this work, we share with [34] the principle of using an over-approximation of the model to rule out some violation of the specification. However, we consider here a different formalism, and we work on discrete samples. In terms of expressiveness of the over-approximated model: *i*) our approach can be seen as less expressive than [34], in the sense that we have a single (uncertain) dynamics, as opposed to LHAs, where a different dynamics can be defined in each mode; this also allows us to propose a simpler (therefore more efficient) analysis, as each new sample allows us to restart from an exact basis, while in [34] at each new sample, the system (from an algorithmic point of view) can be in “different modes at the same time”; *ii*) conversely, our dynamics is also significantly more expressive than the LHA dynamics of [34]; we consider not only the class of linear dynamical systems, but even fit into a special case of non-linear systems, by allowing *uncertainty* in the model dynamics—this is what makes our model an over-approximation of the actual behavior. In addition, we also allow for *uncertain* logs, coping with sensor uncertainties—not considered in [34]. We also propose a new *ad-hoc* implementation based on [18].

In [25,26], a monitor is constructed from a system model in differential dynamic logic [28]. The main difference between [25,26] and our approach relies in the system model: in [25,26], the compliance between the model and the behavior is checked at runtime, while our model is assumed to be an over-approximation of the behavior—which is by assumption compliant with the model.

*Reachability in linear dynamical systems* In [2], given a continuous time linear system with input, the system is discretized and reachable sets for consecutive time intervals are computed. At each step, the *state transition matrix* is expressed using the *Peano-Baker* series. The series is then numerically approximated iteratively using *Riemann sums*. Then a zonotope-based convex hull is computed over-approximating the result of all possible matrices in the uncertain matrix. In [11], Combastel and Raka extend an existing algorithm based on zonotopes so that it can efficiently propagate structured parametric uncertainties. As a result, they provide an algorithm for computation of envelopes enclosing the possible states and/or outputs of a class of uncertain linear dynamical systems. In [22], given an uncertain linear dynamical system  $\dot{x} = A_u x$ , Lal *et al.* provide a sampling interval  $\delta > 0$ , given an  $\epsilon > 0$ , s.t. the piecewise bilinear function, approximating the solution by interpolating at these sample values, is within  $\epsilon$  of the original trajectory. [16] identifies a class of uncertainties by a set of sufficient conditions on the structure of the dynamics matrix  $A_u$ . For such classes of uncertainties, the exact reachable set of the linear dynamical

system can be computed very efficiently. But this method is not applicable for arbitrary classes of uncertainties. In [18], given an uncertain linear dynamical system, we provide two algorithms to compute reachable sets. The first method is based on perturbation theory, and the second method leverages a property of linear systems with inputs by representing them as Minkowski sums. In [17], given an uncertain linear dynamical system, we provide an algorithm to compute statistically correct over-approximate reachable sets using *Jeffries Bayes Factor*. Note that uncertain linear dynamical systems are a special subset of non-linear systems. Thus, uncertain linear dynamical systems can also be modelled as a non-linear system. Some additional works that deal with computing reachable sets of non-linear systems are [8,30,10,14,1,21,9].

### 3 Preliminaries

Formal analysis of safety critical systems requires a precise mathematical model of the system, such as linear dynamical systems. But in reality, the precise, exact model is almost never available—parameter variations, sensor and measurement errors, unaccounted parameters are few such causes that make the availability of a precise model impossible. Presence of such uncertainties in the model makes the safety analysis of these systems, using traditional methods, useless. Thus, for the analysis to be indeed useful, the safety analysis must consider all possible uncertainties. In [22], the authors provide a model, known as *uncertain linear dynamical systems*, to capture such uncertainties. Consider the following example of an uncertain linear dynamical system.

*Example 1 ([16, Example 1.1]).* Let a discrete linear dynamical system  $x^+ = Ax$ , where  $A = \begin{bmatrix} 1 & \alpha \\ 0 & 2 \end{bmatrix}$  and  $\alpha$  represents either the modeling uncertainty or a parameter, assuming  $2 \leq \alpha \leq 3$ . Note that any safety analysis assuming a *fixed* value of  $\alpha$  will render the analysis useless—for the safety analysis to be indeed sound, it must consider *all* possible values of  $\alpha$ , and they cannot be enumerated.

Intuitively, uncertain linear dynamical systems model the uncertainties in the system by representing all possible dynamics matrices of the system—clearly, this forms a special class of non-linear dynamical systems. To perform safety analysis of uncertain linear dynamical systems, these works provide reachable set computation techniques that account for all possible uncertainties.

**Definition 1 (Uncertain linear dynamical systems ([16, Definition 2.4])).** An uncertain linear dynamical system is denoted as

$$x^+ = Ax \tag{1}$$

where  $A \subset \mathbb{R}^{n \times n}$  is the uncertain dynamics matrix.

**Definition 2 (Reachable set of an uncertain linear dynamical systems ([16, Definitions 2.3 and 2.4])).** Given an initial set  $\theta_0$  and time step  $t \in \mathbb{Z}$ , the reachable set of an uncertain linear dynamical system is defined as:

$$RS(A, \theta_0, t) = \theta_t = \{ \theta \mid \theta = \xi_A(\theta_0, t), A \in \Lambda \}. \tag{2}$$

where  $\xi_A(\theta_0, t) = A^t \theta_0$ . An alternative definition is:

$$RS(\Lambda, \theta_0, t) = \theta_t = \bigcup_{A \in \Lambda} \xi_A(\theta_0, t). \quad (3)$$

Note that uncertain linear dynamical systems are capable of modelling systems with parameters or when the system dynamics is not perfectly known—the system has modelling uncertainties. [22,16,18,17] propose various algorithms to compute reachable sets of these systems that account for uncertainties. In this work, we leverage a recently proposed reachable set computation technique, given in [18], to propose our offline and online monitoring algorithm, primarily due to its efficiency vis-à-vis our setting.

Given an initial set  $\theta_0 \subset \mathbb{R}^n$  and given a time step  $t$ , we denote by  $\theta_t \subset \mathbb{R}^n$  the reachable set of the system (given by Eq. (1)) at time step  $t$ . Next, we define a log of the system with uncertainties.

**Definition 3 (Log).** *Given an uncertain linear dynamical system as in Eq. (1), a finite length (uncertain) log of the system is defined as follows:  $\ell = \{(\hat{\theta}_t, t) \mid \theta_t \subseteq \hat{\theta}_t, t \leq H\}$  where  $H$  is a given time bound.*

Each tuple  $(\hat{\theta}_t, t)$  is called a *sample*. Observe that our samples are not necessarily reduced to a *point*. The length of log  $\ell$ —number of samples in  $\ell$ —is given by  $|\ell|$ . Given a log  $\ell$ , the  $k$ -th sample of  $\ell$  is given as  $\ell_k = (\hat{\theta}_{t_k}, t_k)$ , where  $\hat{\theta}_{t_k}$  is an over-approximation of the system at time step  $t_k$ . Note that the length of a log is not necessarily equal to  $H$ , but  $|\ell| \leq H$ : therefore, our logs are *scattered*, in the sense that they do not necessarily contain a sample for each  $t \in \{1, \dots, H\}$ . We further note that the uncertainties in the logs, arising from the sensor uncertainties of the logging system, are independent of the uncertainties in the system modelling (Definition 1). We assume that each sample of the log contains the true state of the system at a given time step. Note that this generally holds in practice: the physical sensors (such as used in medical devices, cars, etc.) record values within an error tolerance, thus giving a range of values containing the actual value.

We call a log  $\ell$  *accurate* if it satisfies the following condition:  $\forall 1 \leq k \leq |\ell| : \hat{\theta}_{t_k} = \theta_{t_k}$ . Given an uncertain linear dynamical system,  $x^+ = \Lambda x$  with an initial set  $\theta_0 \subset \mathbb{R}^n$ , an *over-approximate reachable set of  $x^+$  at time step  $t$*  is  $\text{overReach}(\Lambda, \theta_0, t)$ , such that  $\theta_t \subseteq \text{overReach}(\Lambda, \theta_0, t)$ . We use the technique proposed in [18] to compute  $\text{overReach}(\Lambda, \theta_0, t)$  in this work.

## 4 Monitoring using uncertain linear dynamical systems as bounding model

In this section, we propose the two main contributions of this work: 1) *Offline monitoring*: Given a log with uncertainties, we propose an algorithm to infer the safety of a system as given in Eq. (1). We prove our method’s soundness. 2) *Online monitoring*: We propose a framework to infer safety of a system,

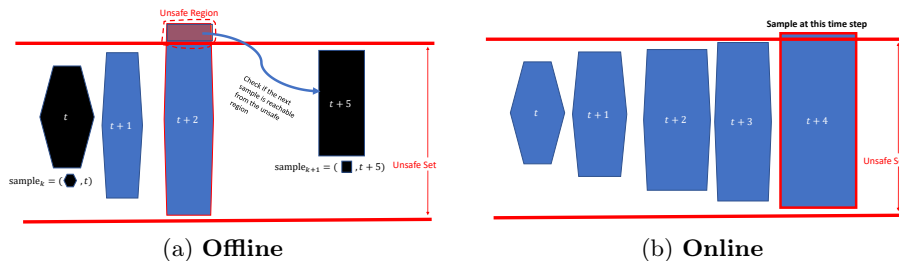


Fig. 2: (2a): **Offline Monitoring**. Black: Two consecutive samples,  $k$  and  $k + 1$ , at time steps  $t$  and  $t + 5$  respectively. Blue: The over-approximate reachable set computed from sample  $k$  using `overReach(.)`. (2b): **Online Monitoring**. Blue: Over-approximate reachable set computed, at each step, using `overReach(.)`.

as in Eq. (1), that triggers the logging system to sample only when needed. Note that, as we only consider the system at *discrete* time steps, the method cannot be sound nor complete, i.e., there always exists a small possibility that the system might violate the safety specification in between two concrete samples (this will be discussed in Section 6). However, our online method *is* both *sound* and *complete at the discrete time stamps*, and under the assumption that the samples are free from uncertainties. That is, our method infers the system to be *safe* if and only if the actual behavior of the system is safe at any discrete time stamp, when the logging system can generate accurate samples of the system. Put it differently, we guarantee that skipping some logging in the future using our method will not remove any sample where a violation could have been observed.

#### 4.1 Offline monitoring

Our first contribution addresses offline monitoring: in this setting, we assume full knowledge of the uncertain log, usually after an execution is completely over. Before we propose our offline algorithm, we illustrate the approach in Fig. 2a. Consider two consecutive samples  $k$  and  $k + 1$ , marked in black, at time steps  $t$  and  $t + 5$  respectively. The reachable sets, in blue, represent the over-approximate behaviors possible by the system between time steps  $t$  and  $t + 5$ . Consider the case where at time step  $t + 2$  the over-approximate reachable set intersects with the unsafe region. Once our algorithm detects a possible unsafe behavior, it computes the intersection between the over-approximate reachable set (here, the reachable set at time-step  $t + 2$ ) and the unsafe set. Then it checks whether the reachable set, given in the next sample ( $k + 1$ ), is reachable from the unsafe region—if yes, it infers *unsafe*; if not, it infers *safe*. Now, we formally propose our offline monitoring method in Algorithm 1 for a given log  $\ell$  with uncertainty.

*Description* The **for** loop, starting in line 1, traverses through each sample, and checks if the system can reach a possibly unsafe behavior between two consecutive samples (computed in lines 2 and 3), using over-approximate reachable

**Algorithm 1:** Offline monitoring

---

```

input  : An uncertain log  $\ell$  of a system  $x^+ = Ax$ , and an unsafe set  $\mathcal{U}$ .
output : Return safe (resp. unsafe) if the actual system behavior is safe
          (resp. potentially unsafe).
1 for  $k \in \{1, \dots, |\ell| - 1\}$  do
2    $(\hat{\theta}_{t_k}, t_k) \leftarrow \ell_k$  ; // current sample
3    $(\hat{\theta}_{t_{k+1}}, t_{k+1}) \leftarrow \ell_{k+1}$  ; // next sample
4    $t_\Delta = t_{k+1} - t_k - 1$  ; // time gap between two samples
5   for  $p \in \{1, \dots, t_\Delta - 1\}$  do
6     if  $\hat{\theta}_{t_k+p} \cap \mathcal{U} \neq \emptyset$  then
7        $\psi \leftarrow \hat{\theta}_{t_k+p} \cap \mathcal{U}$  ; // compute the unsafe region of the system
8        $t_d = t_{k+1} - (t_k + p)$  ;
9        $\vartheta \leftarrow \text{overReach}(A, \psi, t_d)$  ;
          /* Check if next sample is reachable from unsafe */
10      if  $\vartheta \cap \hat{\theta}_{t_{k+1}} \neq \emptyset$  then
11        return unsafe ; // next sample is reachable from unsafe
12       $\hat{\theta}_{t_{k+p+1}} \leftarrow \text{overReach}(A, \hat{\theta}_{t_k+p}, 1)$  ;
13 return safe ;

```

---

set computation. If the over-approximate reachable set between two consecutive samples intersect with the unsafe set (line 6), we perform a refinement as follows (line 7–line 11): We compute the unsafe region (intersection between unsafe set and over-approximate reachable set) in line 7, then check if we can reach the next sample from the unsafe region (line 9–line 11). If the next sample is reachable from the unsafe behavior, we conclude the system is unsafe (line 10–line 11).

*Soundness and incompleteness* Our proposed offline monitoring approach is *sound* at discrete time steps, but not *complete*—there might be cases where our algorithm returns *unsafe* even though the actual system is *safe*. The primary reason for its incompleteness is due to the fact that `overReach(.)` computes an over-approximate reachable set. Formally:

**Theorem 1 (soundness at discrete time steps).** *If the actual system is unsafe at some discrete time step, then Algorithm 1 returns unsafe. Equivalently, if Algorithm 1 returns safe, then the actual system is safe at every discrete step.*

*Proof.* Let the actual trajectory  $\tau$ , between two samples  $k$  and  $k + 1$ , become unsafe at time step  $t_{un}$ . Therefore, the over-approximate reachable set, computed by `overReach(.)` at time step  $t_{un}$ , will also intersect with the unsafe set (due to soundness of `overReach(.)`). Note that the actual trajectory  $\tau$ , originating from the sample  $k$ , intersects the unsafe region at time step  $t_{un}$ , and reaches the sample  $k + 1$ . The refinement module (Algorithm 1, line 7–line 11), using over-approximate reachable sets will therefore infer the same, concluding the system behavior to be unsafe.



## 4.2 Online monitoring

---

**Algorithm 2:** Online monitoring
 

---

```

input  : An uncertain system  $x^+ = Ax$ , an unsafe set  $\mathcal{U}$ , time bound  $H$ .
output : Return safe iff the actual system behavior is safe.
1  $\hat{\theta}_0 \leftarrow$  Sampling at time step 0 ;           // initial behavior of the system.
   /* Check whether the initial behavior is safe */
2 if  $\hat{\theta}_0 \cap \mathcal{U} \neq \emptyset$  then return unsafe ;
3 for  $t \in \{1, 2, \dots, H - 1\}$  do
4    $\hat{\theta}_{t+1} \leftarrow$  overReach( $A, \hat{\theta}_t, 1$ ) ;           // over-approximate reachable set
   /* Check whether the over-approximate reachable set is unsafe */
5   if  $\hat{\theta}_{t+1} \cap \mathcal{U} \neq \emptyset$  then
6      $\ell_{t+1} \leftarrow$  Sample at time step  $t + 1$  ;
7     if  $\ell_{t+1} \cap \mathcal{U} \neq \emptyset$  then
8       return unsafe ;
9      $\hat{\theta}_{t+1} = \ell_{t+1}$  ;                               // reset to actual behavior
10 return safe;

```

---

Given a time bound  $H$ , we propose our online monitoring method in [Algorithm 2](#). The online monitoring algorithm begins by sampling the system at the initial time step, say 0, in [line 1](#). As a sanity check, we confirm if the initial behavior of the system is safe in [line 2](#). The **for** loop starting in [line 2](#)—where each iteration corresponds to the set of actions for a time step  $t$ —performs the following: At a given time step  $t$ , we compute the over-approximate reachable set at the next time step  $t + 1$  ([line 6](#)). If the computed over-approximate reachable set intersects with the unsafe set, we sample the system at time step  $t + 1$  to check if the actual behavior is also unsafe ([line 5–line 9](#)). If safe, we reset the behavior ([line 9](#)); if unsafe, we return *unsafe* ([line 8](#)). Intuitively, this method samples the actual system only when the over-approximate reachable set, computed by `overReach(.)`, intersects the unsafe set. This process is illustrated in [Fig. 2b](#).

*Soundness and completeness* Our online monitoring algorithm is correct (safe and complete) at discrete time steps, *provided* the samples are accurate—it returns safe if and only if the actual behavior of the system is safe at all discrete time steps, when accurate samples are obtained. Intuitively, we get the completeness from the fact that it returns unsafe if and only if the (accurate) sample is unsafe. Formally:

**Theorem 2 (correctness at discrete time steps).** *Algorithm 2 returns safe iff the actual behavior at all discrete time steps is safe.*

*Proof.* The soundness proof—if the actual behavior is unsafe, [Algorithm 2](#) infers unsafe—is straightforward. Hence, we now argue the completeness—if the actual behavior is safe, [Algorithm 2](#) infers *safe*. Note that, [Algorithm 2](#) infers the system behavior as *unsafe* only when a sampled log (actual behavior) becomes unsafe: therefore, if the samples are free from uncertainties (i.e., exact), [Algorithm 2](#) is complete.

*Remark 1.* While our aim is to consider continuous systems, note that, for *discrete-time systems*, our approach is entirely correct (sound and complete), without the restriction to “discrete time steps”, since we can find a granularity small enough for the discrete-time evolution. This is notably the case for systems where the behavior does not change faster than a given frequency (e.g., the processor clock).

## 5 Case studies

We demonstrate the applicability and usability of our approach on two examples, a medical device and an adaptive cruise control. We implemented our online and offline monitoring algorithms in a Python-based prototype tool `MoULDyS`. Tool, models and raw results are available through a GitHub repository<sup>3</sup>. All our experiments were performed on a Lenovo ThinkPad Mobile Workstation with i7-8750H CPU with 2.20 GHz and 32 GiB memory on Ubuntu 20.04 LTS (64 bit). Our tool uses `numpy`, `scipy`, `mpmath` for matrix multiplications, [18] to compute `overReach(.)`, and the `Gurobi` engine for visualization of the reachable sets.

*Implementation details vis-à-vis Algorithms 1 and 2* The intersection checking between two sets in [Algorithms 1](#) and [2](#) has been implemented as an optimization formulation in `Gurobi`. That is, given two sets, our implementation of intersection check returns true iff the two sets intersect. In other words, our intersection check is *exact*. In contrast, *computing* the result of the intersection between two sets adds an over-approximation in our implementation—given two sets, we compute a box hull of the two sets and then compute intersection of the two box hulls. Therefore, the only over-approximate operation we perform in [Algorithms 1](#) and [2](#)—apart from `overReach(.)`—is [Algorithm 1](#) line 7.

*Generating scattered uncertain logs for offline monitoring* At each time step, the logging system may take a snapshot of the system evolution at that time step; the logging occurs with a probability  $p$  (given). In other words, at each time step, it records the evolution of the system with probability  $p$ . Clearly, due to the probabilistic logging, this logger is not guaranteed to generate periodic samples. We also do not assume that the samples logged by the logging system, at each time step, are accurate—the logging system, due to sensor uncertainties, logs an over-approximate sample of the system at that time step. In our experiments, each log was generated statically from our bounding model (the uncertain linear

<sup>3</sup> <https://github.com/bineet-coderep/MoULDyS>

dynamical system) by simulating its evolution from an uncertain initial set (i.e., not reduced to a point). In the end, we get an uncertain log (as in [Definition 3](#)).

*Logging system for online monitoring* When the logging system is triggered, at a time step, to generate a sample, the logging system records the evolution of the system and sends it to the online monitoring algorithm. Similar to the offline logging system, we do not assume that the samples logged by the logging system are perfectly accurate. Here, all the generated logs are *safe*.

*Research questions* We consider the following research questions:

1. Effect of logging probability (number of log samples) on the rate of false alarms raised by the offline monitoring—inferring a behavior as “potentially unsafe” when the actual behavior is “safe”.
2. For offline monitoring, does the size of the samples (in other words, volume of the set obtained as sample), gathered at each step, have an impact on the rate of false alarms? Put it differently, what is the effect, *vis-à-vis* false alarms, of the amount of the uncertainty in the log?
3. For online monitoring, how frequent is the logging system triggered to generate a sample?
4. For the same execution, how do the outcome (in terms of verdict on safety by the monitoring algorithms) and the efficiency (in terms of number of samples needed) of the offline and online monitoring algorithms compare?

### 5.1 First benchmark: Anesthesia

We first demonstrate our approach on an automated anesthesia delivery model [15]. The anesthetic drug considered in this model is propofol. Such safety critical systems are extremely important to be verified formally before they are deployed, as under or overdose of the anesthetic drug can be fatal to the patient.

**Model:** The model as in [15] has two components: 1) Pharmacokinetics (PK): models the change in concentration of the drug as the body metabolizes it. 2) Pharmacodynamics (PD): models the effect of drug on the body. The PK component is further divided into three compartments: *i*) first peripheral compartment  $c_1$ , *ii*) second peripheral compartment  $c_2$ , *iii*) plasma compartment  $c_p$ . The PD component has one compartment, called  $c_e$ . The set of state variables of this system is  $[c_p \ c_1 \ c_2 \ c_e]^\top$ . The input to the system is the infusion rate of the drug (propofol)  $u$ . The complete state-space model of this system is given in [15, Equation 5].

**Model parameters:** The evolution of states— $c_p$ ,  $c_1$ ,  $c_2$ —is dependent on several parameters, such as: the weight of the patient (*weight*), the first order rate constants between the compartments  $k_{10}$ ,  $k_{12}$ ,  $k_{13}$ ,  $k_{21}$  and  $k_{31}$ . The evolution of the state  $c_e$  is dependent on the parameter  $k_d$ , the rate constant between plasma and effect site.

**Safety:** The system is considered safe if the following concentration levels are maintained at all time steps:  $c_p \in [1, 6]$ ,  $c_1 \in [1, 10]$ ,  $c_2 \in [1, 10]$ ,  $c_e \in [1, 8]$ .

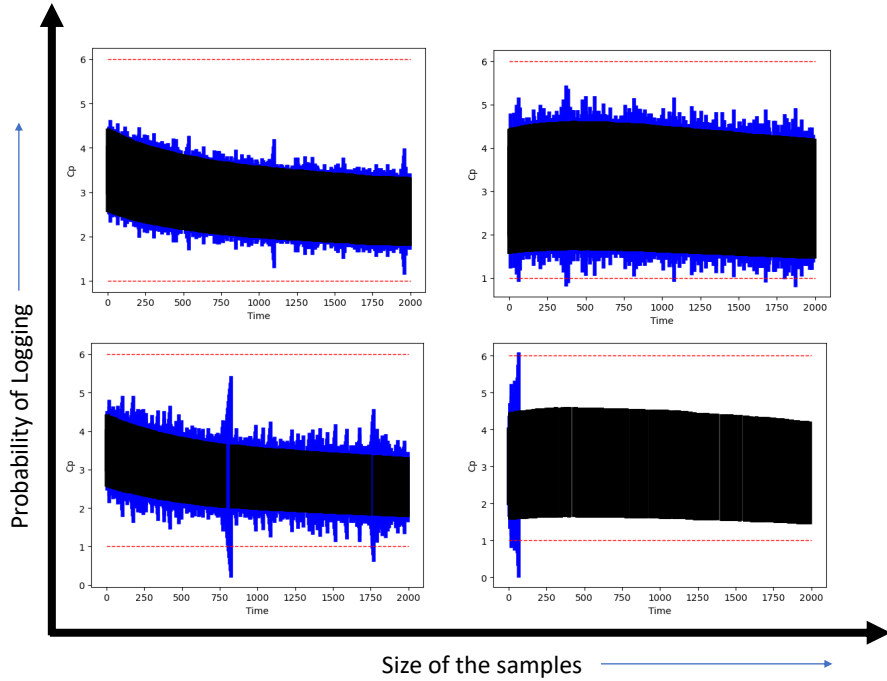


Fig. 3: *Offline monitoring*: We plot the change in concentration level of  $c_p$  with time. The volume of the samples increase from left to right, and the probability of logging increases from bottom to top. The blue regions are the reachable sets showing the over-approximate reachable sets as computed by the offline monitoring, the black regions are the samples from the log given to the offline monitoring algorithm, and the red dotted line represents safe distance level. Note that although the top-row plots and the bottom left plots' reachable sets seem to intersect with the red line (unsafe set), the refinement module infers them to be *unreachable*, therefore concluding the system behavior as *safe*—unlike the bottom-right plot.

In this case study, we focus our attention on the effect of perturbation, in the weight of the patient (*weight*), on the concentration level of plasma compartment  $c_p$ . We assume that the weight of the patient has an additive perturbation of  $\pm 0.8$  kg in this case study—at each time step, the weight of the patient is  $weight + \delta_w$ ,  $\delta_w \in [0, 0.8]$ . With perturbation in the weight, we want to infer safety of this system using monitoring.

We now answer questions (1)-(4), using Figs. 3 and 4. In Fig. 3: *i*) the plots in the bottom row have logging probability of 20%, and the plots in top row have a logging probability of 40%; *ii*) the plots in left column and the right column have been simulated with an initial set of  $[[3,4] [3,4] [4,5] [3,4]]^T$ ,  $u \in [2, 5]$  and

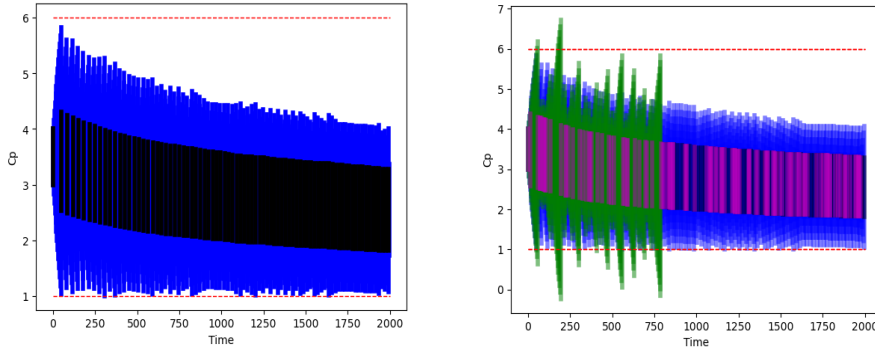


Fig. 4: *Online monitoring*: We plot the change in concentration level of  $c_p$  with time. The blue regions are the reachable sets showing the over-approximate reachable sets as computed by the online monitoring, the black regions are the samples generated when the logging system was triggered by the online monitoring algorithm, and the red dotted line represents safe concentration levels. *Left*: We apply our online monitoring to the anesthesia model. *Right*: We compare our online and offline algorithms. The green regions are the reachable sets showing the over-approximate reachable sets between two consecutive samples from the offline logs, the magenta regions are the offline logs, given as an input to the offline monitoring algorithm, generated by the logging system, and the red dotted line represents safe concentration levels. The blue regions are the reachable sets showing the over-approximate reachable sets as computed by the online monitoring, the black regions are the samples generated when the logging system was triggered by the online monitoring algorithm, and the red dotted line represents safe concentration levels.

$[2,4] [3,6] [3,6] [2,4]^\top$ ,  $u \in [2, 10]$  respectively. That is, the volume of the samples increases from left to right. In Fig. 4, we simulated the trajectory with an initial set  $[3,4] [3,4] [4,5] [3,4]^\top$ ,  $u \in [2, 5]$ .

*Answer to Question 1.* We answer this question by comparing two sets figures in the left column and the right column of Fig. 3. *For the left column, i.e., with smaller sample size*: the bottom-left plot took 51.40s and concluded the system to be safe. The analysis in this plot invoked the refinement module of the offline algorithm. But increasing the probability of logging, i.e., more number of samples, as in the top-left plot, resulted in not invoking the refinement module at all, thus taking 32.92s. *For the right column, i.e., with larger sample size*: this analysis, as shown in the bottom-right column, took 1.73s to complete, and concluded the system behavior to be unsafe. The behavior of the system, shown in top-right plot with 40% probability of logging, results in inferring the behavior of the system as safe, by invoking the refinement module several times.

Overall, this analysis, as shown in the top-right plot, took 35.93s to complete, and concluded the system behavior to be safe.

*Answer to Question 2.* We answer this question by comparing two sets figures in the top row and the bottom row of Fig. 3. *For the bottom row, i.e., with smaller logging probability:* Increasing the volume of the samples results in inferring the behavior from safe (bottom-left plot) to unsafe (bottom-right plot), as per the offline monitoring algorithm. *For the top row, i.e., with higher logging probability:* Increasing the volume of the samples results in not invoking the refinement module (top-left plot) to invoking the refinement module several times (top-right plot), as per the offline monitoring algorithm.

*Answer to Question 3.* The result is given in Fig. 4 (left). Using our online algorithm, we were able to prove safety of the system in 109.04s. The online algorithm triggered the logging system to generate samples for 83 time steps—this is less than 5% of total time steps. We observe, as shown in Fig. 4 (left), that the logging system is triggered more when the trajectory is closer to the unsafe region.

*Answer to Question 4.* We compare our offline and online algorithms, for 2000 time steps, on the same trajectory. The result is given in Fig. 4 (right). Note that, using our online algorithm, we were able to prove safety of the system in 107.99s. The online algorithm triggered the logging system to generate samples only 84 times. In contrast, the offline algorithm, with a log size of 115 (5% logging probability) stopped at the 35th sample, (wrongly) inferring the system as unsafe, taking 71.37s.

## 5.2 Second benchmark: Adaptive Cruise Control

We now apply our algorithms to an adaptive cruise control (ACC) [27]. An ACC behaves like an ordinary cruise control when there is no car in the sight of its sensor, and when there is a car in its sight, it maintains a safe distance.

**Model:** The model as in [27] has the following state variables: *i*) velocity of the vehicle  $v$ , *ii*) distance between the two vehicles  $h$ , and *iii*) velocity of the lead vehicle  $v_L$ . The state space of the system is given in [27, Equation 3]. The set of state variables of this system is  $[v \ h \ v_L]^\top$ .

**Model parameters:** The model is dependent on two parameters: *i*) acceleration of the lead vehicle  $a_L$ , and *ii*) breaking force and torque applied to the wheels as a lumped net force  $F$ . Note that the model is dependent of acceleration of the vehicle  $a_L$ , which is very hard to accurately measure due to sensor uncertainties. Similarly the torque  $F$  applied to the wheels is also dependent of the coefficient of friction of the ground. To reflect such uncertainties, we consider  $a_L \in [-0.9, 0.6]$  and  $F \in [-0.6, 2.46]$ .

**Safety:** The system is safe if the distance between vehicles  $h > 0.5$ .

Consider an event of a car crash, where the log stored by the car before the crash, is the only data available to analyze the crash; such an analysis might benefit police, insurance companies, vehicle manufacturers, etc. Using our offline algorithm one can figure out if the car might have shown unsafe behavior or not. Similarly, consider a vehicle on a highway with a lead vehicle in its sight. The

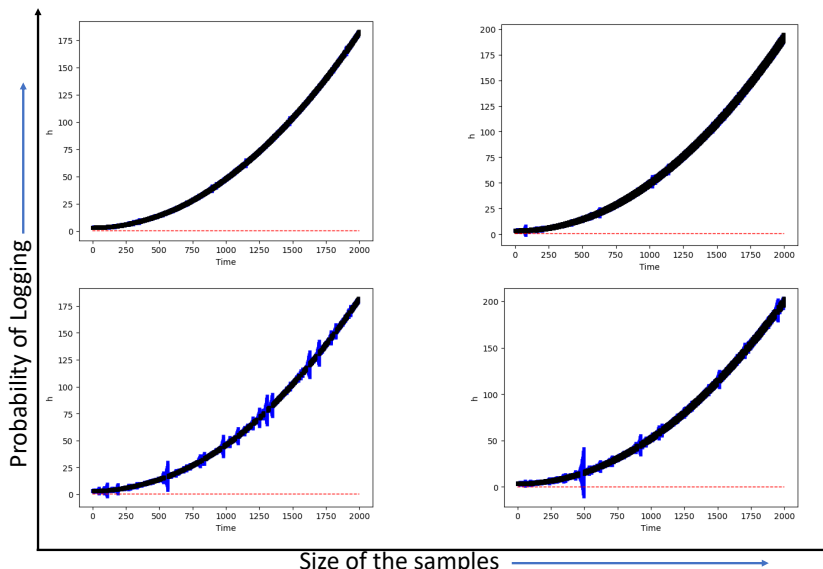


Fig. 5: *Offline monitoring*: We plot the change in distance  $h$  between the vehicles with time. The volume of the samples increase from left to right, and the probability of logging increases from bottom to top.

ACC in such a case needs to continuously read sensor values to track several parameters, such as acceleration of the lead vehicle, braking force, etc.—this results in wastage of energy. Using our online monitoring algorithm, the car reads sensor values only when there is a potential unsafe behavior. This intermittent behavior will result in saving energy without compromising safety of the system.

Next, we answer questions (1)-(4), using Figs. 5 and 6. In Fig. 5: *i*) the plots in the bottom row have logging probability of 20%, and the plots in top row have a logging probability of 40%; *ii*) the plots in left column and the right column have been simulated with an initial set of  $[15, 15.01] [3, 3.03] [14.9, 15]]^T$  and  $[15, 15.1] [3, 3.5] [14.9, 15.1]]^T$  respectively. In Fig. 4, we simulated the trajectory with an initial set  $[15, 15.01] [3, 3.03] [14.9, 15]]^T$ ,  $u \in [2, 5]$ .

*Answer to Question 1.* We answer this question by comparing two sets figures in the left column and the right column of Fig. 5. *For the left column, i.e., with smaller sample size*: the bottom-left plot took 19.08s and concluded the system to be safe. This analysis in this plot invoked the refinement module of the offline algorithm. But increasing the probability of logging, i.e., more number of samples, as in the top-left plot, resulted in not invoking the refinement module at all, thus taking 16.5s. *For the right column, i.e., with larger sample size*: The analysis is similar to that of the left column. The bottom-right plot invoked the refinement module several times, thus taking 20.84s, while the top-right plot took 17.5s, as it invoked the refinement module a smaller number of times.

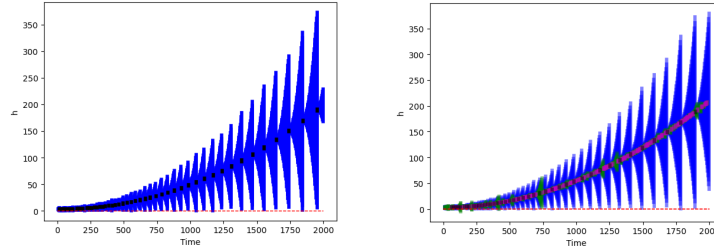


Fig. 6: *Online monitoring*: We plot the change in distance between two vehicle  $h$  with time. The color coding is same as Fig. 4. *Left*: We apply our online monitoring to the ACC model. *Right*: We compare our online and offline algorithms.

*Answer to Question 2.* We answer this question by comparing two sets figures in the top row and the bottom row of Fig. 5. *For the bottom row, i.e., with smaller logging probability*: Comparing the bottom-left and bottom-right shows that increasing sample volume results in invoking the refinement module more frequently. A very similar behavior is seen by comparing the top row (i.e., with higher logging probability).

*Answer to Question 3.* Using our online algorithm, we were able to prove safety of the system in 104.58s. The online algorithm triggered the logging system to generate samples for 53 time steps—this is less than 3% of total time steps. This is shown in Fig. 6 (left).

*Answer to Question 4.* We compare our offline and online algorithm, for 2000 time steps, on the same trajectory. The result is given in Fig. 6 (right). Note that, using our online algorithm, we were able to prove safety of the system in 124.46s. The online algorithm triggered the logging system to generate samples only 50 times. In contrast, the offline algorithm, with a log size of 281 (14% logging probability) took 28.54s to infer that the system is safe.

### 5.3 General observations

In this section, we provide general answers to questions (1)-(4):

*Answer to Question 1.* Increasing the probability of logging reduces the chances of inclusion of spurious behaviors due to over-approximate reachable set computation over longer time horizon. Therefore, it has a reduced chance of spuriously inferring the system unsafe, also fewer chance of invoking the refinement module (as there are less spurious behaviors).

*Answer to Question 2.* Increasing the size of samples (due to uncertainties or inherent nature of the system) results in increasing chances of invoking the refinement module more frequently. It also increases the chance of (wrongly) inferring the system to be unsafe, as the refinement module can in itself add to the overapproximation.



*Answer to Question 3.* We observed that our online algorithm is able to prove the system’s safety very efficiently with very few samples.

*Answer to Question 4.* We observed that for a given random log, the offline algorithm was unable to prove safety of the system, whereas our online algorithm was able to prove safety of the system, using fewer samples, by intelligently sampling the system only when needed. We also note that, though here we just demonstrated the result for one random log, but our internal experiments showed that the online algorithm always needed fewer samples to prove safety—which is unsurprising, as it is designed to sample the system only when needed. This can also result in energy saving, as sampling usually requires energy and bandwidth.

*Reachable sets computation using Flow\** As uncertain linear dynamical systems are a special type of non-linear systems, Flow\* [8] would have been a natural candidate to benchmark our offline and online monitoring implementation by comparing various methods to compute `overReach(·)`. However, we ran into the following issues: *i)* To the best of our understanding, Flow\* expects the model of the continuous dynamics to be given as input, along with a discretization parameter. Therefore, trying to encode the time-varying uncertainties in the system as state variables will lead to discretization of the variables encoding uncertainties; such discretization leads to undesired behavior, as those uncertain variables will fail to capture the actual range of values that are possible at any time step. *ii)* However, Flow\* does allow time varying uncertainties, but only additive<sup>4</sup>. Unfortunately, our benchmark requires *multiplicative* uncertainties. Still, we believe Flow\* could be compared with our implementation when the bounding model has a simpler dynamics than our uncertain linear dynamical systems.

## 6 Conclusion

We presented a new approach for monitoring cyber-physical systems against safety specifications, using the additional knowledge of an over-approximation of the system expressed using an uncertain linear dynamic system. Our approach assumes as first input a log with exact (but scattered) timestamps and uncertain variable samplings (in the form of zonotopes), and as second input an over-approximated model, bounding the possible behaviors. The over-approximation is modeled by *uncertainty* in the variables of the dynamics. In the offline setting, we are thus able to detect possible violations of safety properties, by extrapolating the known samples with the over-approximated dynamics, and if needed using a second reachability analysis to check whether the next sample is “compatible” with the possible unsafe behavior, i.e., can be reached from the unsafe zone. In the online setting, we are capable of *decreasing* the number of samples, triggering a sample only when there might be a safety violation in a near future, based on the latest known sample and on the over-approximated model dynamics—increasing the energetic efficiency. Our method is sound in the sense

<sup>4</sup> See example at <https://flowstar.org/benchmarks/2-dimensional-ltv-system/>

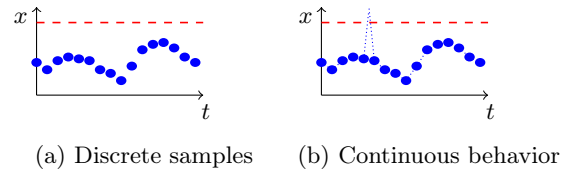


Fig. 7: Incompleteness

that an absence of detection of violation by our method indeed guarantees the absence of an actual violation at any discrete time step. In the online method, provided the samples are accurate, our method is in addition complete, i.e., the method outputs *safe* iff the actual system is safe at all discrete time steps. Put it differently, we guarantee that *not* triggering a sample at some time steps is harmless and will not lead to missing a safety violation.

*Future works.* On the log side, we considered fixed timestamps, but uncertain values for the continuous variables; in fact, the timestamps could also be uncertain. This makes sense when the samples are triggered by sensors distributed over a network, which can create delays and therefore timed uncertainty. This was not considered in our approach, and is on our agenda.

A possible threat to validity remains the *enumeration* of time steps in both our algorithms (line 5 in Algorithm 1 and line 3 in Algorithm 2), which could slow down the analysis for very sparse logs—even though this did not seem critical in our experiments. Using skipping methods could help improving the efficiency of our approach.

Another future work consists in increasing our guarantees, notably due to the *continuous* nature of cyber-physical systems under monitoring. Indeed, even with a rather fine-grained sampling showing no specification violation (e.g., in Fig. 7a), it can always happen that the actual *continuous* behavior violated the specification (e.g., in Fig. 7b). While setting discrete time steps at a sufficiently fine-grained scale will help to increase the confidence in the results of our approach, no absolutely formal guarantee can be derived. Therefore, one of our future works is to propose some additional conditions for extrapolating (continuous) behaviors between consecutive discrete samples. Also, improving the scope of our guarantees (in the line of, e.g., [12]) is on our agenda.

## References

- [1] Althoff, M.: An introduction to CORA 2015. In: ARCH@CPSWeek. EPiC Series in Computing, vol. 34, pp. 120–151. EasyChair (2015). <https://doi.org/10.29007/zbkv>
- [2] Althoff, M., Le Guernic, C., Krogh, B.H.: Reachable set computation for uncertain time-varying linear systems. In: HSCC. pp. 93–102. ACM (2011). <https://doi.org/10.1145/1967701.1967717>

- [3] André, É., Hasuo, I., Waga, M.: Offline timed pattern matching under uncertainty. In: ICECCS. pp. 10–20. IEEE Computer Society (2018). <https://doi.org/10.1109/ICECCS2018.2018.00010>
- [4] Bakhirkin, A., Ferrère, T., Nickovic, D., Maler, O., Asarin, E.: Online timed pattern matching using automata. In: FORMATS. LNCS, vol. 11022, pp. 215–232. Springer (2018). [https://doi.org/10.1007/978-3-030-00151-3\\_13](https://doi.org/10.1007/978-3-030-00151-3_13)
- [5] Bartocci, E., Deshmukh, J.V., Donzé, A., Fainekos, G.E., Maler, O., Nickovic, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In: Lectures on Runtime Verification – Introductory and Advanced Topics, LNCS, vol. 10457, pp. 135–175. Springer (2018). [https://doi.org/10.1007/978-3-319-75632-5\\_5](https://doi.org/10.1007/978-3-319-75632-5_5)
- [6] Basin, D.A., Klaedtke, F., Zalinescu, E.: The MonPoly monitoring tool. In: RV-CuBES. Kalpa Publications in Computing, vol. 3, pp. 19–28. EasyChair (2017)
- [7] Becchi, A., Zaffanella, E.: Revisiting polyhedral analysis for hybrid systems. In: SAS. LNCS, vol. 11822, pp. 183–202. Springer (2019). [https://doi.org/10.1007/978-3-030-32304-2\\_10](https://doi.org/10.1007/978-3-030-32304-2_10)
- [8] Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow\*: An analyzer for non-linear hybrid systems. In: CAV. LNCS, vol. 8044, pp. 258–263. Springer (2013). [https://doi.org/10.1007/978-3-642-39799-8\\_18](https://doi.org/10.1007/978-3-642-39799-8_18)
- [9] Chen, X., Sankaranarayanan, S.: Decomposed reachability analysis for non-linear systems. In: RTSS. pp. 13–24. IEEE Computer Society (2016). <https://doi.org/10.1109/RTSS.2016.011>
- [10] Chen, X., Sankaranarayanan, S., Ábrahám, E.: Under-approximate flowpipes for non-linear continuous systems. In: FMCAD. pp. 59–66. IEEE (2014). <https://doi.org/10.1109/FMCAD.2014.6987596>
- [11] Combastel, C., Raka, S.A.: On computing envelopes for discrete-time linear systems with affine parametric uncertainties and bounded inputs. IFAC Proceedings Volumes **44**(1), 4525 – 4533 (2011). <https://doi.org/10.3182/20110828-6-IT-1002.02585>
- [12] Daurer, J.C., Finkbeiner, B., Schirmer, S.: Monitoring with verified guarantees. In: Feng, L., Fisman, D. (eds.) RV. LNCS, vol. 12974, pp. 62–80. Springer (2021). [https://doi.org/10.1007/978-3-030-88494-9\\_4](https://doi.org/10.1007/978-3-030-88494-9_4)
- [13] Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: CAV. LNCS, vol. 8044, pp. 264–279. Springer (2013). [https://doi.org/10.1007/978-3-642-39799-8\\_19](https://doi.org/10.1007/978-3-642-39799-8_19)
- [14] Duggirala, P.S., Mitra, S., Viswanathan, M., Potok, M.: C2E2: A verification tool for stateflow models. In: TACAS. LNCS, vol. 9035, pp. 68–82. Springer (2015). [https://doi.org/10.1007/978-3-662-46681-0\\_5](https://doi.org/10.1007/978-3-662-46681-0_5)
- [15] Gan, V., Dumont, G.A., Mitchell, I.: Benchmark problem: A PK/PD model and safety constraints for anesthesia delivery. In: ARCH@CPSWeek. EPIc Series in Computing, vol. 34, pp. 1–8. EasyChair (2014). <https://doi.org/10.29007/8drm>
- [16] Ghosh, B., Duggirala, P.S.: Robust reachable set: Accounting for uncertainties in linear dynamical systems. ACM Transactions on Embedded Computing Systems **18**(5s), 97:1–97:22 (2019). <https://doi.org/10.1145/3358229>
- [17] Ghosh, B., Duggirala, P.S.: Reachability of linear uncertain systems: Sampling based approaches. Tech. Rep. 2109.07638, arXiv (2021), <https://arxiv.org/abs/2109.07638>
- [18] Ghosh, B., Duggirala, P.S.: Robustness of safety for linear dynamical systems: Symbolic and numerical approaches. Tech. Rep. 2109.07632, arXiv (2021), <https://arxiv.org/abs/2109.07632>

- [19] Halbwachs, N., Proy, Y.É., Raymond, P.: Verification of linear hybrid systems by means of convex approximations. In: SAS. LNCS, vol. 864, pp. 223–237. Springer (1994). [https://doi.org/10.1007/3-540-58485-4\\_43](https://doi.org/10.1007/3-540-58485-4_43)
- [20] Jakšić, S., Bartocci, E., Grosu, R., Nguyen, T., Ničković, D.: Quantitative monitoring of STL with edit distance. FMSD **53**(1), 83–112 (2018). <https://doi.org/10.1007/s10703-018-0319-x>
- [21] Kong, S., Gao, S., Chen, W., Clarke, E.M.: dReach:  $\delta$ -reachability analysis for hybrid systems. In: TACAS. LNCS, vol. 9035, pp. 200–205. Springer (2015). [https://doi.org/10.1007/978-3-662-46681-0\\_15](https://doi.org/10.1007/978-3-662-46681-0_15)
- [22] Lal, R., Prabhakar, P.: Bounded error flowpipe computation of parameterized linear systems. In: EMSOFT. pp. 237–246. IEEE (2015). <https://doi.org/10.1109/EMSOFT.2015.7318279>
- [23] Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: FORMATS and FTRTFT. LNCS, vol. 3253, pp. 152–166. Springer (2004). [https://doi.org/10.1007/978-3-540-30206-3\\_12](https://doi.org/10.1007/978-3-540-30206-3_12)
- [24] Mamouras, K., Chattopadhyay, A., Wang, Z.: A compositional framework for quantitative online monitoring over continuous-time signals. In: RV. LNCS, vol. 12974, pp. 142–163. Springer (2021). [https://doi.org/10.1007/978-3-030-88494-9\\_8](https://doi.org/10.1007/978-3-030-88494-9_8)
- [25] Mitsch, S., Platzer, A.: ModelPlex: verified runtime validation of verified cyber-physical system models. FMSD **49**(1-2), 33–74 (2016). <https://doi.org/10.1007/s10703-016-0241-z>
- [26] Mitsch, S., Platzer, A.: Verified runtime validation for partially observable hybrid systems. Tech. rep. (2018), <http://arxiv.org/abs/1811.06502>
- [27] Nilsson, P., Hussien, O., Balkan, A., Chen, Y., Ames, A.D., Grizzle, J.W., Ozay, N., Peng, H., Tabuada, P.: Correct-by-construction adaptive cruise control: Two approaches. IEEE Transactions on Control Systems Technology **24**(4), 1294–1307 (2016). <https://doi.org/10.1109/TCST.2015.2501351>
- [28] Platzer, A.: The complete proof theory of hybrid systems. In: LICS. pp. 541–550. IEEE Computer Society (2012). <https://doi.org/10.1109/LICS.2012.64>
- [29] Qin, X., Deshmukh, J.V.: Clairvoyant monitoring for signal temporal logic. In: FORMATS. LNCS, vol. 12288, pp. 178–195. Springer (2020). [https://doi.org/10.1007/978-3-030-57628-8\\_11](https://doi.org/10.1007/978-3-030-57628-8_11)
- [30] Testylier, R., Dang, T.: NLTOOLBOX: A library for reachability computation of nonlinear dynamical systems. In: ATVA. LNCS, vol. 8172, pp. 469–473. Springer (2013). [https://doi.org/10.1007/978-3-319-02444-8\\_37](https://doi.org/10.1007/978-3-319-02444-8_37)
- [31] Ulus, D., Ferrère, T., Asarin, E., Maler, O.: Timed pattern matching. In: FORMATS. LNCS, vol. 8711, pp. 222–236. Springer (2014). [https://doi.org/10.1007/978-3-319-10512-3\\_16](https://doi.org/10.1007/978-3-319-10512-3_16)
- [32] Waga, M., Akazaki, T., Hasuo, I.: A Boyer-Moore type algorithm for timed pattern matching. In: FORMATS. LNCS, vol. 9884, pp. 121–139. Springer (2016). [https://doi.org/10.1007/978-3-319-44878-7\\_8](https://doi.org/10.1007/978-3-319-44878-7_8)
- [33] Waga, M., André, É.: Online parametric timed pattern matching with automata-based skipping. In: NFM. LNCS, vol. 11460, pp. 371–389. Springer (2019). [https://doi.org/10.1007/978-3-030-20652-9\\_26](https://doi.org/10.1007/978-3-030-20652-9_26)
- [34] Waga, M., André, É., Hasuo, I.: Model-bounded monitoring of hybrid systems. ACM Transactions on Cyber-Physical Systems (2022), to appear
- [35] Waga, M., André, É., Hasuo, I.: Parametric timed pattern matching. ACM Transactions on Software Engineering and Methodology (2022), to appear