

# Model-Bounded Monitoring of Hybrid Systems\*

Masaki Waga<sup>1,2</sup> , Étienne André<sup>3</sup> , and Ichiro Hasuo<sup>2,4</sup> 

<sup>1</sup>Kyoto University, Japan

<sup>2</sup>National Institute of Informatics, Japan

<sup>3</sup>Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

<sup>4</sup>The Graduate University for Advanced Studies, Japan

## Abstract

Monitoring of hybrid systems attracts both scientific and practical attention. However, monitoring algorithms suffer from the methodological difficulty of only observing sampled discrete-time signals, while real behaviors are continuous-time signals. To mitigate this problem of sampling uncertainties, we introduce a *model-bounded monitoring* scheme, where we use prior knowledge about the target system to prune interpolation candidates. Technically, we express such prior knowledge by linear hybrid automata (LHAs)—the LHAs are called *bounding models*. We introduce a novel notion of *monitored language* of LHAs, and we reduce the monitoring problem to the membership problem of the monitored language. We present two partial algorithms—one is via reduction to reachability in LHAs and the other is a direct one using polyhedra—and show that these methods, and thus the proposed model-bounded monitoring scheme, are efficient and practically relevant.

**Keywords**— monitoring, cyber-physical systems, hybrid automata

\*This is the author (and slightly extended) version of the manuscript of the same name published in the proceedings of the 12th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPs 2021). The final version is available at [dl.acm.org](https://dl.acm.org). This work is partially supported by JST ACT-X Grant No. JPMJAX200U, by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), by JSPS Grant-in-Aid No. 18J22498, and by ANR-NRF ProMiS (ANR-19-CE25-0015).

## 1 Introduction

**Monitoring** Pervasiveness and safety criticalness of *cyber-physical systems (CPSs)*—where physical dynamics are controlled by software—pose their quality assurance as a pressing industrial and social problem. A number of research efforts have aimed at their correctness proofs, with software science and control theory collaborating hand-in-hand.

However, such *exhaustive verification* is often very hard with real-world examples. This is because *white-box models* of real-world CPSs are hard to find—the difficulty can be because of 1) the systems’ complexity, 2) uncertainties in their operation environments, and 3) third-party black-box components. Mathematically, formal verification is to give a *proof* that a system is correct, and a white-box model is a *definition* of the system. Without a white-box model, there is no definition to build a proof on.

*Monitoring* is attracting attention as a light-weight yet feasible alternative in quality assurance of CPSs. Monitoring consists in checking whether a sequence of data satisfies a specification expressed using some formalism. It can be used offline (e.g., for extracting interesting parts from a huge log) and online (e.g., for alerting to unsafe phenomena). See the related work in this section for references.

**Hybrid System Monitoring** In this paper, we study monitoring of CPSs, with a particular emphasis on their *hybrid* aspect (i.e., the interplay between continuous and discrete worlds).

We sketch the workflow of hybrid system monitoring in Fig. 1. We are given a specific behavior  $\sigma$  of the system under monitoring (SUM) and a specification  $\varphi$  (in this paper, we focus on safety specifications). The problem is

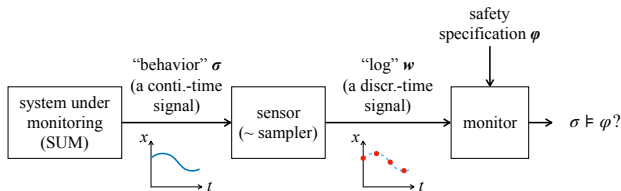
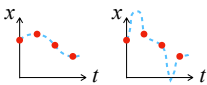


Figure 1: Hybrid system monitoring and sampling uncertainties

to decide whether  $\sigma$  is safe or not, in the sense that  $\sigma$  satisfies  $\varphi$ . We assume a computer solves this problem. Therefore, an input to a monitor must be a discrete-time signal  $w$ , obtained from the continuous-time signal  $\sigma$  via sampling. We shall call such  $w$  a *log* of the SUM induced by the behavior  $\sigma$ .

### Sampling Uncertainties in Hybrid System Monitoring

There is a methodological difficulty already in the high-level schematics in Fig. 1: Figure 2:  $w$  and



*By looking only at a sampled log  $w$ , how can a monitor conclude anything about the real behavior  $\sigma$ ?*

The same log  $w$  can result from different behaviors  $\sigma$ . Fig. 2 shows an example, where we cannot decide if a safety property “ $x$  is always nonnegative” is satisfied by  $\sigma$ . In other words, the way we interpolate the log  $w$  and recover  $\sigma$  is totally arbitrary. Thus, we cannot exclude potential violations of any safety specification unless the specification happens to talk only about values at sampling instants.

This issue of *sampling uncertainties* is often ignored in the hybrid system monitoring literature. They typically employ heuristic interpolation methods, such as piecewise-constant and piecewise-linear interpolation (above). Use of these heuristic interpolation methods is often justified, typically when the sampling rate is large enough. However, in *networked monitoring* scenarios where a sensor and a monitor are separated by, e. g., a wireless network, the sampling rate is small, and the interpolation of a log becomes a real issue. Network monitoring is increasingly common in IoT applications, and smaller sampling rates (i. e., longer sampling intervals) are preferred for energy efficiency.

**Example 1** (automotive platooning). Consider a situation where two vehicles drive one after the other, with their distance kept small. Such *automotive platooning* at-

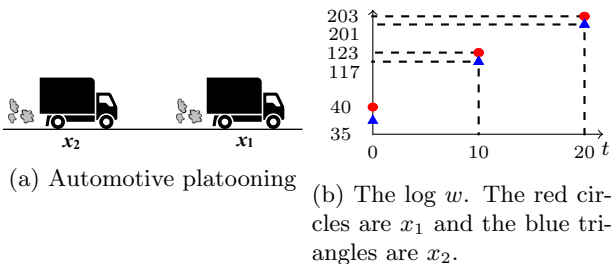


Figure 3: A leading example: automotive platooning

tracts interest as a measure for enhanced road capacity as well as for fuel efficiency (by reducing air resistance).

Assume that the monitoring is conducted on a remote server. Each vehicle intermittently sends its position to the server via the Internet. Thus, the remote monitor only has a coarse-grained log. Concretely, a log  $w$  is given in Fig. 3b, by the position  $x_1, x_2$  (meters) of each of the two vehicles, sampled at time  $t = 0, 10, 20$  (seconds).

Let us now ask this question: *have the two vehicles touched each other?* Physical contact of the vehicles is not observed in Fig. 3b, but we cannot be sure what happened between the sampling instants. The piecewise-constant and piecewise-linear interpolation can only answer to this question approximately. Moreover, such approximation is not of much help in the current example where sampling intervals are long.

### Interpolation Assisted by System Knowledge

The following idea underpins the current work.

*Prior knowledge about a system is a powerful tool to bound sampling uncertainties.*

The latter means excluding some candidates when we recover a behavior  $\sigma$  from a word  $w$  by interpolation (cf. Fig. 2). For the log in Fig. 3b, for example, we can say  $x_1$  never reached  $10^4$ , knowing that the vehicle cannot accelerate that quickly.

Putting this idea to actual use requires a careful choice of a knowledge representation formalism.

- For one, it is desired to be *expressive*. The above “acceleration rate” argument can be formulated in terms of Lipschitz constants, but it is nice to also include mode switching—an important feature of hybrid systems.
- For another, a formalism should be *computationally tractable*. Monitoring is a practice-oriented method that often tries to process a large amount of data with limited computing resources (especially in embedded applications). Therefore, inference over knowledge

represented in the chosen formalism should better be efficient.

Note that these two concerns—expressivity and computational tractability—are in a trade-off.

**Bounding Models Given by LHAs** In this paper, we express such prior knowledge about a system using a *linear hybrid automaton (LHA)* [HPR94]. This LHA is called a *bounding model*, and serves as an overapproximation of the target system.

LHA is one of the well-known subclasses of hybrid automata (HA); an example is in Fig. 6a. LHA’s notable simplifying feature is that flow dynamics is restricted to a conjunction of linear (in)equalities over the derivatives  $\dot{x}_1, \dot{x}_2, \dots, \dot{x}_M$ . Its expressivity is limited—for example, a flow specification  $\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{b}$  is not allowed since the variables  $\mathbf{x}$  occur there. Differential inclusions are allowed, nevertheless (such as  $\dot{x}_1 \in [7.5, 8.5]$  and  $\dot{x}_1 - \dot{x}_2 \leq 1$ ); these are useful in expressing known safety envelopes, as in Fig. 6a. Most importantly, analysis of LHAs is tractable, with convex polyhedra providing an efficient means to study the reachability problem.

**Model-Bounded Hybrid System Monitoring** Our proposal is a scheme that we call *model-bounded monitoring* of hybrid systems. Its workflow is in Fig. 4; its features are as follows.

1. We use our prior knowledge about the SUM in order to reduce sampling uncertainties. The knowledge is expressed by an LHA; it is called a *bounding model*.
2. We restrict to a safety specification  $\varphi$  given by a conjunction of linear (in)equalities.<sup>1</sup> We interpret  $\varphi$  globally (“ $\sigma(t)$  satisfies  $\varphi$  at any time  $t$ ”). We obtain an LHA  $\mathcal{M}_{\neg\varphi}$  by taking the synchronized product of the bounding model  $\mathcal{M}$  and an automaton monitoring the violation of  $\varphi$ .
3. We introduce the notion of *monitored language*  $\mathcal{L}_{\text{mon}}$  of an LHA. Roughly speaking, it is the set of “logs which have a corresponding signal accepted by the LHA.” The notion differs from known language notions for LHA, in that mode switches in an LHA need not be visible in a log (modes may change between sampling instants).
4. We show the following meta-level correctness result:  $w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$  if and only if there exists a continuous-time signal  $\sigma$  such that

<sup>1</sup>This restriction is for the ease of presentation. Extension to LTL specifications should not be hard: an LTL formula can be translated to an automaton; and it can then be combined with a bounding model  $\mathcal{M}$ . This is future work.

- (a)  $\sigma$  induces  $w$  by sampling,
- (b)  $\sigma$  conforms with the bounding model  $\mathcal{M}$ , and
- (c)  $\sigma$  violates the safety specification  $\varphi$ .

Our main technical contribution consists of

1. the introduction of the new language notion  $\mathcal{L}_{\text{mon}}$ ,
2. the use of  $\mathcal{L}_{\text{mon}}$  in the proposed model-bounded monitoring scheme, and
3. (partial) algorithms that solve  $\mathcal{L}_{\text{mon}}$  membership.

Used in the scheme in Fig. 4, these algorithms check if the given log  $w$  belongs to  $\mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ , whose answer is then used for the safety analysis of the (unknown) actual behavior  $\sigma$ . The last point is discussed in the next paragraph about usage scenarios.

We present two (partial) algorithms: one reduces the  $\mathcal{L}_{\text{mon}}$  membership problem to the reachability problem of LHAs, translating a log  $w$  into an LHA. The other is a direct algorithm that relies on polyhedra computation. These algorithms are necessarily partial since  $\mathcal{L}_{\text{mon}}$  membership is undecidable (Theorem 17). However, their positive and negative answers are guaranteed to be correct. Moreover, we observe that the latter direct algorithm terminates in most benchmarks, especially when a bounding model’s dimensionality is not too large.

**Example 2.** We continue Example 1. For the log  $w$  in Fig. 3b, the bounding model  $\mathcal{M}$  in Fig. 6a confines potential interpolation between the samples to the hatched areas in Fig. 5. The two areas are separate in  $t \in [0, 10]$ , which means the two cars were safe in the period. For  $t \in [10, 20]$ , the two areas overlap, suggesting potential collision.

The above analysis is automated by our automata-theoretic framework in Fig. 4. We shall sketch its workflow.

Let  $\varphi$  be the safety specification  $x_1 - x_2 > 0$  (“no physical contact”)<sup>2</sup>. The formal construction of  $\mathcal{M}_{\neg\varphi}$  (Definition 12) yields the LHA in Fig. 6b. In  $\mathcal{M}_{\neg\varphi}$ , the original LHA  $\mathcal{M}$  (Fig. 6a) is duplicated, and once  $\varphi$  is violated, the execution can move from the first copy (the top two states in Fig. 6b) to the second (the bottom states). The bottom states are accepting—they detect violation of  $\varphi$ .

Now we use one of our algorithms to solve the membership problem, i.e., whether the log  $w$  belongs to  $\mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ . Solving this membership problem amounts

<sup>2</sup>For simplicity, we modeled the cars as points. It is straightforward to use a more realistic model e.g., a car model by a rectangle.

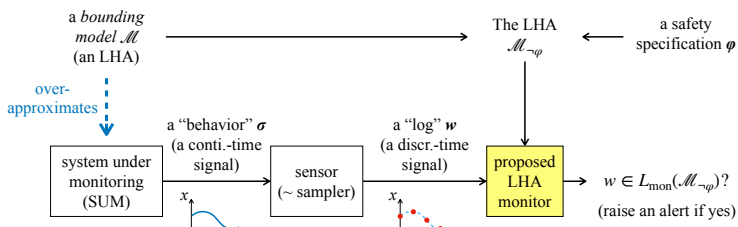


Figure 4: Model-bounded monitoring of hybrid systems

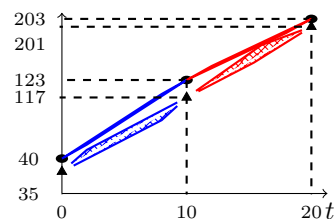
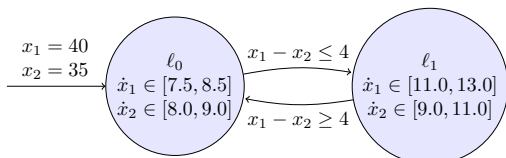
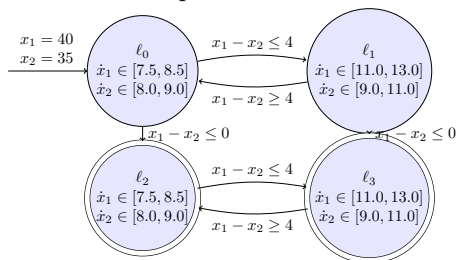


Figure 5: Model-bounded monitoring of the log  $w$  in Fig. 3b. The bounding model  $\mathcal{M}$  in Fig. 6a confines interpolation to the hatched area. Thus no collision in  $t \in [0, 10]$ ; potential collision in  $t \in [10, 20]$ .



(a) A bounding model  $\mathcal{M}$  for the platooning example, expressed as an LHA



(b) The LHA  $\mathcal{M}_{\neg\varphi}$  for  $\varphi = (x_1 - x_2 > 0)$

Figure 6: LHAs for the automotive platooning example

to computing the hatched areas in Fig. 5—it is done relying on polyhedra computation—and checking if the safety specification is violated.

**Usage Scenarios** The scheme in Fig. 4 is used as follows. As a basic prerequisite, we assume that the bounding model  $\mathcal{M}$  *overapproximates* the SUM: for each continuous-time signal  $\sigma$ ,

*(soundness of a bounded model)*

$\sigma$  is a behavior of the SUM  $\implies \sigma$  is a run of  $\mathcal{M}$ .

We do not require the other implication. Due to the limited expressivity of LHAs (that is the price for computational tractability),  $\mathcal{M}$  would not tightly describe the SUM.

Assume first that our monitor did *not* raise an alert (i. e.,  $w \notin \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ ). Let  $\sigma_0$  be the (unknown) actual behavior of the SUM that is behind the log  $w$ . By the feature 4 of the scheme, we conclude that  $\sigma_0$  was safe. Indeed,  $\sigma_0$  satisfies Item 4a by definition. It comes from the SUM, and thus by the soundness assumption,  $\sigma_0$  satisfies Item 4b. Hence Item 4c must fail.

Let us turn to the case where our monitor *did* raise an alert ( $w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ ). This can be a false alarm. For one, the existence of unsafe  $\sigma$  (as in the feature 4) does not imply that the actual behavior  $\sigma_0$  was unsafe. For another, Item 4b does not guarantee that  $\sigma$  is indeed a possible behavior of the SUM, since we only assume *soundness* of the bounding model. Nevertheless, a positive answer of our monitor comes with a reachability witness (a trace) in  $\mathcal{M}_{\neg\varphi}$ , which serves as a useful clue for further examination.

Summarizing, our monitor’s alert can be false, while the absence of an alert proves safety. We can thus say our model-bounded monitoring scheme is *sound*.

**Bounding Models** We note that the roles of bounding models are different from common roles played by *system models*. A system model aims to describe the system’s behaviors in a *sound* and *complete* manner. In contrast, bounding models focus on overapproximation, trading completeness for computational tractability that is needed in monitoring applications.

The *overapproximating* nature of a bounding model is less of a problem in monitoring, compared to other exhaustive applications such as model checking. In the latter, approximation errors accumulate over time, leading to increasingly loose overapproximation. In contrast, in our usage, a bounding model is used to interpolate between samples (Fig. 5). Here overapproximation errors get reset to zero by new samples.

Because we assume that we have an overapproximation of the actual system, we can indeed formally verify the safety of the system by the reachability analysis of the bounding model. However, due to the overapproximation, the given LHA usually contain unsafe behaviors, i. e., the unsafe locations are reachable in the LHA. Most of the benchmarks in our experiments are certainly unsafe. In contrast, in model-bounded monitoring, even if the unsafe locations are reachable in the LHA, the monitored behavior can be safe, i. e., the unsafe locations are unreachable from the current sample. Therefore, monitoring is still useful even if we have a model overapproximating the actual system.

Bounding models can arise in different ways, including:

- **(Adding margins to a system model)** If a system model is given as an LHA, we can use it as a bounding model. A more realistic scenario is to add some margins to address potential perception and actuation errors. LHAs’ feature that they allow differential *inclusions* is particularly useful here. An example is in Fig. 7, where perception and actuation uncertainties are addressed by the additional margins in the transition guards and flow dynamics, respectively.
- **(LHA approximation of a system model)** LHA is one of the subclasses of HA for which reachability is attackable (it is hopeless for general HA). Consequently, tools have been proposed for analyzing LHA, including PHAVerLite [BZ19] and its predecessor PHAVer [Fre08]. Moreover, for their application, overapproximation of other dynamics by LHAs has been studied and tool-supported. See e. g., [Fre08, Section 3.2]. These techniques can be used to obtain an LHA bounding model from a more complex model.
- **(From a third-party vendor)** HA is a well-accepted formalism in academia and industry. It is conceivable that a system vendor provides an LHA as the system’s “safety specification”. It serves as a bounding model.

**Contributions** We summarize our main contributions.

- We tackle the issue of sampling uncertainties in hybrid system monitoring, proposing the *model-bounded monitoring* scheme (Fig. 4) as a countermeasure. The scheme uses LHAs as *bounding models*.
- We introduce the novel technical notion of *monitored language*  $\mathcal{L}_{\text{mon}}$  for LHAs. In  $\mathcal{L}_{\text{mon}}$ , unlike in other language notions, input words and mode switches do not necessarily synchronize. We show that  $\mathcal{L}_{\text{mon}}$

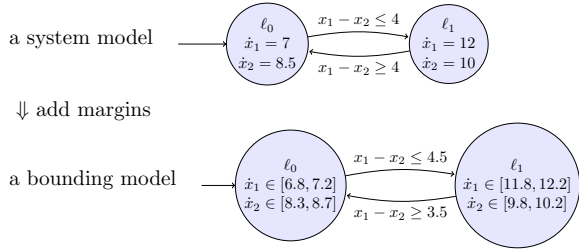


Figure 7: *Adding margins* to obtain bounding models. The top model gets loosened by perception uncertainties (margin 0.5) and actuation uncertainties (margin 0.2)

membership is undecidable, yet we introduce two partial algorithms.

- We establish soundness of our model-bounded monitoring scheme: absence of an alert guarantees that every possible behavior  $\sigma$  behind the log  $w$  is safe.
- The practical relevance and algorithmic scalability is demonstrated by experiments, using benchmarks that are mainly taken from automotive platooning scenarios.

**Related Work** In the IoT applications [Gub+13], energy efficiency is of paramount importance. Energy efficiency demands longer sampling and communication intervals; the current work presents an automatic and sound method to mitigate the uncertainties that result from those longer intervals.

In the context of quality assurance of CPSs, monitoring of *digital* (i.e., discrete-valued) or *analog* (i.e., continuous-valued) signals takes an important role. There have been many works on signal monitoring using various logic e.g., *signal temporal logic (STL)* [MN04; FP09], *timed regular expressions (TREs)* [Ulu+14], *timed automata* [Bak+18], or *timed symbolic weighted automata (TSWAs)* [Wag19]. However, in most of the existing works, interpolation of the sampled signals is limited to only piecewise-constant or piecewise-linear.

There are a few works on monitoring utilizing system models. In [ZLD12], a set of predictive words are generated through a static analysis of the monitored program and monitored against linear temporal logic. In [Pin+17], the system model and the monitored property are given as timed automata to construct a monitor predicting the satisfaction (or violation) of the monitored property. In [BGF18], the stochastic system model is trained as a hidden Markov model and utilized for monitoring against a DFA. In [QD20], statistical models on the monitored

signals are utilized to predict the future signals for robust monitoring against signal temporal logic.

Overall, prediction (i.e., *extrapolation*) of the future behaviors is the main purpose of the existing model-based monitoring works [ZLD12; Pin+17; BGF18; QD20] to the best of our knowledge. Our approach utilizes system knowledge for *interpolation* of the infrequently sampled signals.

There are existing language notions for LHAs [AKV98]. These are different from the notion  $\mathcal{L}_{\text{mon}}$  that we introduce; hence the results in [AKV98] do not subsume ours. The key difference is if the input word and mode switches must synchronize; see Example 11 and the preceding discussions.

A recent line of work is that of timed pattern matching [Ulu+14; WHS17; Bak+18; UM18; AHW18; WA19; WAH19; Wag19], that takes as input a log and a specification, and decides *where* in the log the specification is satisfied or violated. Through the construction of the *matching automata* [Bak+18; Wag19], our monitoring problem can also decide where in the log the specification is satisfied or violated. Thus, our work can also be seen as an extension of timed pattern matching concerning the system models.

**Outline** We recall LHAs in Section 2. After we introduce monitored languages  $\mathcal{L}_{\text{mon}}$  for LHAs in Section 3, model-bounded monitoring is formalized in Section 4 and we prove its correctness. We show that  $\mathcal{L}_{\text{mon}}$  membership is undecidable in Section 5. We present two partial algorithms:

1. the one in Section 6 relies on an existing model checker PHAVerLite via suitable translation; and
2. the one in Section 7 is a dedicated algorithm.

We perform extensive experiments in Section 8 and conclude in Section 9.

## 2 Preliminaries: Linear Hybrid Automata

Let  $\mathcal{I}(\mathbb{Q})$  be the set of closed intervals on  $\mathbb{Q}$ , i.e., of the form  $[a, b]$ , where  $a, b \in \mathbb{Q}$  and  $a \leq b$ . For a partial function  $f: X \rightarrow Y$ , the domain  $\{x \in X \mid f(x) \text{ is defined}\}$  is denoted by  $\text{dom}(f)$ . We fix a set  $\mathbb{X} = \{x_1, \dots, x_M\}$  of real-valued *variables*. A (*variable*) *valuation* is a function  $v: \mathbb{X} \rightarrow \mathbb{R}$ . When  $\mathbb{X}$  is clear from the context, a valuation  $v$  is expressed by the tuple  $(v(x_1), v(x_2), \dots, v(x_M))$ . Given  $\mu: \mathbb{X} \rightarrow \mathcal{I}(\mathbb{Q})$ , we define the *update* of a valuation  $v$ , written  $[v]_\mu$ , as follows:  $[v]_\mu(x) \in \mu(x)$  if  $x \in \text{dom}(\mu)$ , and  $[v]_\mu(x) = v(x)$  otherwise.

We assume  $\bowtie \in \{\leq, =, \geq\}$ . Let  $\Phi(\mathbf{x})$  be the set of *linear systems* over  $\mathbb{X}$  defined by a finite conjunction of inequalities of the form  $a_1x_1 + a_2x_2 + \dots + a_Mx_M \bowtie d$ , with  $d, a_1, a_2, \dots, a_M \in \mathbb{Z}$ . We let  $\top = \bigwedge \emptyset$  and  $\perp$  be the contradiction. The set  $\Phi(\dot{\mathbf{x}})$  is defined similarly; it consists of constraints over derivatives  $\dot{x}_1, \dots, \dot{x}_M$ .

## 2.1 Syntax

**Definition 3** (linear hybrid automata (LHA) [HPR94]). An LHA is a tuple  $\mathcal{M} = (L, F, \mathbb{X}, \text{Init}, \mathcal{F}, \text{Inv}, E)$ , where:

1.  $L$  is a finite set of locations,
2.  $F \subseteq L$  is the set of accepting locations,
3.  $\mathbb{X}$  is a finite set of variables,
4.  $\text{Init}: L \rightarrow \Phi(\mathbf{x})$  is the initial variable valuation for each location,
5.  $\mathcal{F}: L \rightarrow \Phi(\dot{\mathbf{x}})$  is the *flow*, assigning to each  $\ell \in L$  the set of the derivatives (“rates”)  $\{(\dot{x}_1, \dot{x}_2, \dots, \dot{x}_M) \mid (\dot{x}_1, \dot{x}_2, \dots, \dot{x}_M) \models \mathcal{F}(\ell)\}$ ,
6.  $\text{Inv}: L \rightarrow \Phi(\mathbf{x})$  is the *invariant* for each location,
7.  $E$  is a finite set of *edges*  $e = (\ell, g, \mu, \ell')$  where
  - (a)  $\ell, \ell' \in L$  are the source and target locations,
  - (b)  $g \in \Phi(\mathbf{x})$  is the guard,
  - (c)  $\mu: \mathbb{X} \rightarrow \mathcal{I}(\mathbb{Q})$  is the update function.

Note that **Definition 3** allows for non-deterministic initial locations. A location  $\ell$  that cannot be initial is such that  $\text{Init}(\ell) = \perp$ .

LHAs can be *composed* using synchronized product (see e.g., [Ras05, Definition 4]) in a way similar to finite-state automata. The synchronized product of two LHAs is known to be an LHA [HPR94]. Of importance is that, in a composed location, the global flow constraint is the *intersection* of the local component flow constraints.

**Example 4.** Consider the LHA in **Fig. 6a**, where  $\text{Init}$  is such that  $\text{Init}(\ell_0) = \{x_1 = 40 \wedge x_2 = 35\}$  and  $\text{Init}(\ell_1) = \perp$ . This LHA, giving a bounding model for an automotive platooning system (**Example 1**), contains 2 locations and 2 variables  $\mathbb{X} = \{x_1, x_2\}$ . This LHA features no invariant (i.e., all invariants are  $\top$ ). Note that this LHA fits into a subclass in which the derivatives for the flows are all in bounded, constant intervals.

In this LHA (**Fig. 6a**),  $x_1$  (resp.  $x_2$ ) denotes the position of Vehicle 1 (resp. 2), initially 40 and 35 respectively. In  $\ell_0$ , both vehicles run roughly at the same speed, although Vehicle 2 can be slightly faster (e.g., due to smaller

air resistance, as it follows Vehicle 1). When the distance between both vehicles becomes less than 4, they enter mode  $\ell_1$ , where Vehicle 1 drives faster than in  $\ell_0$ .

In the LHA  $\mathcal{M}_{\neg\varphi}$  in **Fig. 6b**, the vertical edges are enabled once the specification  $x_1 - x_2 > 0$  is violated, that is, once the two vehicles touch each other.

**Example 5.** Consider the LHA in **Fig. 8**, where

1.  $\mathbb{X} = \{x_1, x_2, t_{abs}, t_{rel}\}$ ,
2.  $\text{Init}$  is such that  $\text{Init}(\ell_0) = \{x_1 = 40 \wedge x_2 = 35 \wedge t_{abs} = 0 \wedge t_{rel} = 0\}$  and  $\text{Init}(w\ell_i) = \perp$  for  $1 \leq i \leq 3$ , and
3.  $\dot{x}_1 = \dot{x}_2 = \dot{t}_{abs} = \dot{t}_{rel} = 1$  in all locations (not depicted in **Fig. 8**).

We depict invariants using a box under the location.

## 2.2 Semantics

We recall the standard semantics of LHAs, called *concrete semantics*. It is formulated as a timed transition system [HMP92].

**Definition 6** (concrete semantics of an LHA). Given an LHA  $\mathcal{M} = (L, F, \mathbb{X}, \text{Init}, \mathcal{F}, \text{Inv}, E)$ , the *concrete semantics* of  $\mathcal{M}$  is given by the timed transition system (TTS)  $(S, S_0, \rightarrow)$ , with

- $S = \{(\ell, v) \in L \times \mathbb{R}^M \mid v \models \text{Inv}(\ell)\}$ ,
- $S_0 = \{(\ell, v) \mid \ell \in L, v \in \text{Init}(\ell)\} \cap S$ ,
- $\rightarrow$  consists of the discrete and continuous transition relations:

1. discrete transitions:  $(\ell, v) \xrightarrow{e} (\ell', v')$ , if there exists  $e = (\ell, g, \mu, \ell') \in E$  such that  $v \models g$ ,  $v' \in [v]_\mu$ .
2. continuous transitions:  $(\ell, v) \xrightarrow{d;f} (\ell, v')$ , with the delay  $d \in \mathbb{R}_{\geq 0}$  and the flow  $f: \mathbb{X} \rightarrow \mathbb{R}$  satisfying,  $f \models \mathcal{F}(\ell)$ ,  $\forall d' \in [0, d], (\ell, v + d'f) \in S$ , and  $v' = v + df$ , where  $v + d'f$  is the valuation satisfying  $(v + d'f)(x) = v(x) + d'f(x)$  for any  $x \in \mathbb{X}$ .

**Definition 7** ((accepting) run). Given an LHA  $\mathcal{M}$  with concrete semantics  $(S, S_0, \rightarrow)$ , we refer to the states of  $S$  as the *concrete states* of  $\mathcal{M}$ . A *run* of  $\mathcal{M}$  is an alternating sequence  $\rho = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n$  of concrete states  $s_i \in S$  and transitions  $\rightarrow_i \in \rightarrow$  satisfying  $s_0 \in S_0$  and  $s_0 \rightarrow_1 s_1 \dots \rightarrow_n s_n$ . For a run  $\rho$ , the *duration*  $\text{Dur}(\rho) \in \mathbb{R}_{\geq 0}$  is the sum of the delays in  $\rho$ . We denote the  $i$ -th prefix  $s_0 \rightarrow_1 s_1 \dots \rightarrow_i s_i$  of  $\rho$  by  $\rho[i]$ . A run is *accepting* if its last state  $(\ell, v)$  satisfies  $\ell \in F$ .

**Example 8.** Let  $\mathcal{M}$  be the LHA in Fig. 6a. The sequence  $\rho = (\ell_0, v_0) \xrightarrow{10, (8.3, 8.2)} (\ell_0, v_1) \xrightarrow{\frac{4}{3}, (7.5, 9)} (\ell_0, v_2) \xrightarrow{e_1} (\ell_1, v_2) \xrightarrow{\frac{2}{3}, (12, 9)} (\ell_1, v_3) \xrightarrow{e_2} (\ell_0, v_3) \xrightarrow{8, (7.75, 8.25)} (\ell_0, v_4)$  is a run of  $\mathcal{M}$ , where  $v_0 = (40, 35)$ ,  $v_1 = (123, 117)$ ,  $v_2 = (133, 129)$ ,  $v_3 = (141, 135)$ ,  $v_4 = (203, 201)$ , and  $e_1$  and  $e_2$  are the edges from  $\ell_0$  and  $\ell_1$ , respectively.

### 3 Monitored Languages of LHAs

We introduce another semantics for LHAs besides concrete semantics (Definition 6); it is called the *monitored language*. The two semantics are used in Fig. 4 in the following way:

1. concrete semantics is (roughly) about whether a continuous-time signal  $\sigma$  (“behavior”) conforms with the LHA  $\mathcal{M}$ ;
2. the monitored language  $\mathcal{L}_{\text{mon}}(\mathcal{M})$  is about whether a discrete-time signal  $w$  (“log”) conforms with  $\mathcal{M}$ .

**Definition 9** (timed quantitative words). A *timed quantitative word*  $w$  is a sequence  $(u_1, \tau_1), (u_2, \tau_2), \dots, (u_m, \tau_m)$  of pairs  $(u_i, \tau_i)$  of a valuation  $u_i: \mathbb{X} \rightarrow \mathbb{R}$  and a timestamp  $\tau_i \in \mathbb{R}_{\geq 0}$  satisfying  $\tau_i \leq \tau_{i+1}$  for each  $i \in \{1, 2, \dots, m-1\}$ .

For a timed quantitative word  $w = (u_1, \tau_1), \dots, (u_m, \tau_m)$ , we let  $|w| = m$  and for any  $i \in \{1, \dots, m\}$ , we let  $w[i] = (u_1, \tau_1), \dots, (u_i, \tau_i)$ .

We sometimes refer to pairs  $(u_i, \tau_i)$  as *samples*—these are the red dots in Fig. 4.

**Definition 10** (monitored language  $\mathcal{L}_{\text{mon}}(\mathcal{M})$ ). Let  $\rho = s_0 \rightarrow_1 s_1 \rightarrow_2 \dots \rightarrow_n s_n$  be a run of an LHA  $\mathcal{M}$  (Definition 6), and  $w = (u_1, \tau_1), (u_2, \tau_2), \dots, (u_m, \tau_m)$  be a timed quantitative word. We say  $w$  is *associated* with  $\rho$  if, for each  $j \in \{1, 2, \dots, m\}$ , we have either of the following two. Here  $\ell_i, v_i$  are so that  $s_i = (\ell_i, v_i)$  for each  $i \in \{0, 1, \dots, n\}$ .

1. There exists  $i \in \{0, 1, 2, \dots, n\}$  such that  $\text{Dur}(\rho[i]) = \tau_j$  and  $u_j = v_i$ ; or
2. There exists  $i \in \{0, 1, 2, \dots, n-1\}$  such that  $\text{Dur}(\rho[i]) < \tau_j < \text{Dur}(\rho[i+1])$  and for any  $x \in \mathbb{X}$ ,  $u_j(x) = v_i(x) + (\tau_j - \text{Dur}(\rho[i]))f_i(x)$  holds, where  $\rightarrow_i = \xrightarrow{d_i, f_i}$ .

Finally, the *monitored language*  $\mathcal{L}_{\text{mon}}(\mathcal{M})$  of an LHA  $\mathcal{M}$  is the set of timed quantitative words associated with some accepting run of  $\mathcal{M}$ .

In the above definition of association of  $w$  to  $\rho$ , note that the lengths of  $\rho$  and  $w$  can differ ( $n \neq m$ ). Condition 1 is when a sample in  $w$  happens to be simultaneous with some transition in  $\rho$ . This special case is not required to happen at all, for  $w$  to be associated with  $\rho$ .

For example, in Fig. 5, mode switches (i.e., discrete transitions) in the LHA in Fig. 6a can occur at times other than  $t = 0, 10, 20$ . This is in contrast to the language of hybrid automata in [AKV98], where (observable) discrete transitions are always *synchronous* with the word, much like the condition 1.

**Example 11.** Let  $\mathcal{M}$  be the LHA in Fig. 6a,  $\rho$  be the run of  $\mathcal{M}$  in Example 8. The timed quantitative word  $w$  in Fig. 3b is associated with  $\rho$ . We note that the sampling and the discrete transitions are *asynchronous*: the sampling is at 0, 10, and 20, while the discrete transitions are at  $\frac{34}{3}$  and 12. This is in contrast to the *synchronous* language in [AKV98]: the accepted words represent the discrete transitions e.g., at  $\frac{34}{3}$  and 12.

### 4 The Model-Bounded Monitoring Scheme

Based on the technical definitions in Section 3, we formally introduce the scheme that we sketched in Fig. 4. (Partial) algorithms for computing if  $w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$  are introduced in later sections. Recall that we focus on safety specifications that are global and linear.

**Definition 12** (the LHA  $\mathcal{M}_{\neg\varphi}$ ). Let  $\mathcal{M}$  be an LHA, and  $\varphi \in \Phi(\mathbf{x})$  (Section 2). The LHA  $\mathcal{M}_{\neg\varphi}$  is defined by

- making a copy  $\mathcal{M}^\circ$  of  $\mathcal{M}$ ,
- making each location  $\ell^\circ$  of  $\mathcal{M}^\circ$  non-initial i.e.,  $\text{Init}(\ell^\circ) = \perp$ ,
- letting  $F$  consist of all the states  $\ell^\circ$  of  $\mathcal{M}^\circ$ , and
- for each location  $\ell \in \mathcal{M}$ , creating an edge  $(\ell, \neg\varphi, \emptyset, \ell^\circ)$  from  $\ell$  to its copy  $\ell^\circ$ , labeling the edge with the safety specification  $\varphi$  as a guard and no update.

Fig. 6b shows an example of  $\mathcal{M}_{\neg\varphi}$ . In  $\mathcal{M}_{\neg\varphi}$ , having a single accepting sink state for a violation of  $\varphi$  is not enough. After detecting violation, we are still obliged to check if the rest of a word  $w$  conforms with the bounding model  $\mathcal{M}$ . Thus, we maintain a copy of  $\mathcal{M}$ .



**Lemma 13.** *The following are equivalent for each sequence  $\rho$ .*

- Both of the following hold:
  1.  $\rho$  is a (non-necessarily accepting) run of  $\mathcal{M}$ , and
  2.  $\rho$  violates  $\varphi$  at a certain time instant.
- There is an accepting run  $\rho'$  of  $\mathcal{M}_{\neg\varphi}$  such that
 
$$\{w \mid w \text{ is associated with } \rho\} = \{w \mid w \text{ is associated with } \rho'\}.$$

The proof is easy by definition. The runs  $\rho$  and  $\rho'$  can differ only in the locations they visit—in an LHA, an enabled transition is not always taken. Note, however, that violation of  $\varphi$  and  $w$ 's association are two properties that are insensitive to locations.

We are ready to state the correctness of our scheme (Fig. 4). The proof is straightforward by Lemma 13 and Definition 10.

**Theorem 14** (correctness). *In the setting of Definition 12, let  $w$  be a timed quantitative word. We have  $w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$  if and only if there is a (non-necessarily accepting) run  $\rho$  of  $\mathcal{M}$  such that*

1.  $w$  is associated with  $\rho$ , and
2.  $\rho$  violates  $\varphi$  at some time instant.

Identifying a run  $\rho$  with a behavior  $\sigma$ , and association of  $w$  to  $\rho$  with sampling, the theorem establishes the feature 4 of our scheme (Section 1).

The consequence in the safety analysis of the real SUM (instead of its bounding model  $\mathcal{M}$ ) is discussed in the “usage scenario” paragraph of Section 1. In particular, due to potential gaps between the SUM and the bounding model  $\mathcal{M}$ , an alert of our monitor can be false, while the absence of an alert proves safety. Overall, *our model-bounded monitoring scheme is sound*.

**Example 15.** We show how the illustration in Example 2 is formalized by the monitored language  $\mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$  of the bounded model  $\mathcal{M}_{\neg\varphi}$ . Let  $\mathcal{M}_{\neg\varphi}$  be the LHA in Fig. 6b and  $w$  be the timed quantitative word in Fig. 3b.

We have  $w[2] \notin \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$  since all the runs with which  $w[2]$  is associated are not accepting. That is, the log  $w$  is safe until time  $t = 10$ .

However, for the full log, we have  $w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ , because of the following accepting run  $\rho$  with which  $w$  is associated:  $\rho = [(\ell_0, u_1) \xrightarrow{10, (8.3, 8.2)} (\ell_0, u_2) \xrightarrow{4, (7.5, 9.0)} (\ell_0, v) \xrightarrow{6, (\frac{25}{3}, 8.0)} (\ell_2, u_2)]$ , where  $v = (153, 153)$ .

## 5 Membership for Monitored Languages: Symbolic Interpolation

The rest of the paper is devoted to solving the membership problem of  $\mathcal{L}_{\text{mon}}(\mathcal{M})$ , a core computation task in Fig. 4. We will present two (partial) algorithms: they are symbolic algorithms that iteratively update polyhedra.

### The $\mathcal{L}_{\text{mon}}$ membership problem:

INPUT: An LHA  $\mathcal{M}$  and a timed quantitative word  $w$ .

PROBLEM: Return the set  $\mathcal{C}(w, \mathcal{M})$  of indices  $i$  satisfying  $u[i] \in \mathcal{L}_{\text{mon}}(\mathcal{M})$ . In particular,  $w \in \mathcal{L}_{\text{mon}}(\mathcal{M})$  iff  $|w| \in \mathcal{C}(w, \mathcal{M})$ .

**Example 16.** Let  $\mathcal{M}_{\neg\varphi}$  be the LHA in Fig. 6b and  $w$  be the timed quantitative word in Fig. 3b. We have  $\mathcal{C}(w, \mathcal{M}_{\neg\varphi}) = \{3\}$ , meaning  $w[1], w[2] \notin \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ , and  $w[3] = w \in \mathcal{L}_{\text{mon}}(\mathcal{M}_{\neg\varphi})$ . This result corresponds to the illustration in Fig. 5.

The following “no-go” theorem is sort of expected, given previous results from the hybrid automata literature.

□ **Theorem 17** (undecidability). *For an LHA  $\mathcal{M}$  and a timed quantitative word  $w$ , it is undecidable to decide the emptiness of  $\mathcal{C}(w, \mathcal{M})$ .*

Given the undecidability result, we can think of restricting the class of the models. For example, the problem becomes decidable if the number of discrete transitions within a time unit is bounded [Bu+19].

Nevertheless, in practice, as we observe in Section 8, our partial algorithms below perform effectively for many benchmarks on the full LHA class—especially our latter, direct algorithm.

Our two partial algorithms have the following features.

- Instead of solving one-way reachability (forward or backward) as many existing algorithms do, they solve *interpolation* between two points i.e., samples (see Fig. 5).
- They work in a *one-shot manner*, collecting the linear constraints representing the interpolations from the given origin to the end with only bounded-time forward reachability analysis. Instead, a naive method would iterate between forward and backward reachability analysis.

## 6 Method I: via Reduction to LHA Reachability Analysis

In our first solution, we reduce the  $\mathcal{L}_{\text{mon}}$  membership problem to reachability analysis of LHAs. In practice, we will use PHAVerLite, one of the most efficient tools for reachability analysis of hybrid systems according to [BZ19].

The idea of reducing monitoring to reachability analysis of extensions of finite-state automata is not new and was already proposed in the literature e.g., [AHW18]. While both [AHW18] and the method we introduce in this section are symbolic; the differences are in the formalism and problem. On the one hand, [AHW18] uses parametric timed automata as a parametric specification and performs *parametric* timed pattern matching (which can be seen as parametric monitoring). On the other hand, we use LHAs for the bounding model and we perform symbolic monitoring. An extension for a parametric setting is a future work, which is technically not much demanding.

Our workflow is as follows:

1. We transform the input timed quantitative word  $w$  into an LHA  $\mathcal{M}_w$  (that is in fact only timed, i.e., it only uses clocks), that uses two extra variables:
  - (a)  $t_{\text{abs}}$  measures the absolute time since the beginning of the word; and
  - (b)  $t_{\text{rel}}$  measures the (relative) time since the last sampled timed quantitative word.
2. We perform the synchronized product  $\mathcal{M} \parallel \mathcal{M}_w$  of the given LHA  $\mathcal{M}$  with the transformed LHA  $\mathcal{M}_w$ .
3. We run the reachability analysis procedure for the product LHA  $\mathcal{M} \parallel \mathcal{M}_w$ , to derive all possible locations  $w\ell_i$  of  $\mathcal{M}_w$  such that  $(\ell, w\ell_i)$  is reachable in  $\mathcal{M} \parallel \mathcal{M}_w$  with  $t_{\text{rel}} = 0$ , where  $\ell$  is an accepting location of the given LHA  $\mathcal{M}$ .

We explain these steps in the following.

### 6.1 Transforming the Timed Quantitative Word into an LHA

First; we transform the input timed quantitative word  $w$  into an LHA. The resulting LHA  $\mathcal{M}_w$  is a simple sequence of locations with guarded transitions in between, also resetting  $t_{\text{rel}}$ .

The LHA  $\mathcal{M}_w$  features an absolute time clock  $t_{\text{abs}}$  (initially 0, of rate 1 and never reset), and can test all variables of the system in guards (these are not reset in this

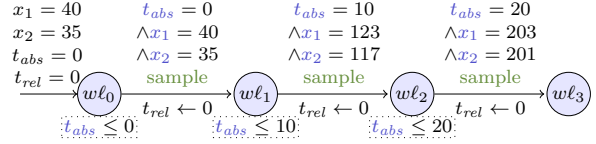


Figure 8: TQW2LHA applied to the timed quantitative word in Fig. 3b. Here, *i*)  $\mathbb{X} = \{x_1, x_2, t_{\text{abs}}, t_{\text{rel}}\}$ , *ii*) Init is such that  $\text{Init}(\ell_0) = \{x_1 = 40 \wedge x_2 = 35 \wedge t_{\text{abs}} = 0 \wedge t_{\text{rel}} = 0\}$  and  $\text{Init}(w\ell_i) = \perp$  for  $1 \leq i \leq 3$ , and *iii*)  $t_{\text{abs}} = t_{\text{rel}} = 1$  and  $\dot{x}_1, \dot{x}_2 \in \mathbb{R}$  in all locations. Invariants are boxed under the location.

LHA, though). More in details, we simply convert each sample  $(u_i, \tau_i)$  of the timed quantitative word  $w$  into a guard of the LHA testing for the timestamp using the absolute time clock  $t_{\text{abs}}$ , and for the value of the variables. The invariant of the location preceding a timestamp  $\tau_i$  also features the clock constraint  $t_{\text{abs}} \leq \tau_i$  (this is not crucial for correctness but limits the state space explosion). The transitions are all labeled with a fresh action **sample** (which could be replaced with an unobservable action, but such actions are not accepted by the PHAVerLite model checker). Each transition resets  $t_{\text{rel}}$ . Overall, this procedure shares similarities with the one that transforms a timed word into a timed automaton, proposed in [AHW18].

Let TQW2LHA denote this procedure.

For example, consider the timed quantitative word  $w$  in Fig. 3b. The result  $\mathcal{M}_w$  of TQW2LHA( $w$ ) is given in Fig. 8.

### 6.2 Reachability Analysis Using PHAVerLite

We perform the synchronized product  $\mathcal{M}_w \parallel \mathcal{M}$  (“parallel composition”) of the LHA  $\mathcal{M}_w$  constructed from  $w$  together with the given LHA  $\mathcal{M}$ .

Then, we run the reachability analysis, setting as target the states for which both of the following conditions hold:

1. the monitor is in an accepting location; and
2.  $t_{\text{rel}} = 0$ .

The latter condition ensures that only the states such that we just sampled a word are accepting. Thanks to the latter condition, we can take into account of the next sample without any explicit backward reachability analysis.

This is intuitively because of the following. While  $t_{\text{rel}} > 0$  and we are at  $w\ell_i$  in  $\mathcal{M}_w$ , we compute all the reachable valuations from  $w[i]$  by the forward reachability

analysis. Here, we non-deterministically make an assumption of the next sample including the ones incompatible with the actual sample  $w[i+1]$ . When we take the transition from  $w\ell_i$  to  $w\ell_{i+1}$  and  $t_{rel} = 0$ , we require that the assumption of the next sample must be compatible with the actual sample  $w\ell_{i+1}$  and accept only if the accepting locations are reachable by an interpolation between  $w[i]$  and  $w[i+1]$ .

**Example 18.** Let us exemplify the need for the latter condition. Consider the LHA  $\mathcal{M}_{-\varphi}$  in Fig. 6b and the timed quantitative word  $w$  in Fig. 3b transformed into the LHA  $\mathcal{M}_w$  in Fig. 8 (only the time frame in  $[0, 10]$  is of interest in this example). Clearly, this log is safe w.r.t. the LHA  $\mathcal{M}_{-\varphi}$  in Fig. 6b, that is neither  $\ell_2$  nor  $\ell_3$  are reachable, since the distance between both vehicles cannot be  $\leq 0$  in the  $[0, 10]$  time frame.

If we simply run the reachability procedure looking for  $\ell_2$  or  $\ell_3$  as target (without condition on  $t_{rel}$ ), the procedure will output that at least  $\ell_2$  is reachable. Indeed, it is possible that vehicle 1 runs at the minimal rate of 7.5 while vehicle 2 runs at the maximal rate of 9. In that case, after 10 time units, vehicle 1 (resp. 2) reaches  $x$ -coordinate 115 (resp. 125), and thus their distance is  $\leq 0$ , making  $\ell_2$  reachable. While this behavior is indeed possible from the knowledge we have of the first sample, it is actually impossible knowing the full log and in particular the second sample. This phenomenon is illustrated in the part of Fig. 5 restricted to the  $[0, 10]$  time frame: the blue part depicts all possible valuations knowing the first and second sample.

Hence, adding the condition  $t_{rel} = 0$  forces the model checker to take into consideration the next sample before making a decision concerning the reachability of a possible target location.

## 7 Method II: Direct Method by Polyhedra Computation

In our second solution, we directly solve the  $\mathcal{L}_{\text{mon}}$  membership problem. We iteratively compute the runs of the LHA  $\mathcal{M}$  associated with the prefixes of the timed quantitative word  $w$  utilizing bounded reachability analysis. This is our main contribution.

Procedure 1 shows an outline of our incremental procedure for the  $\mathcal{L}_{\text{mon}}$  membership problem. Procedure 1 incrementally constructs the intermediate states  $State_i$  and outputs the partial result  $Result_i$  showing if  $w[i] \in$

---

**Procedure 1:** Outline of our incremental procedure for the  $\mathcal{L}_{\text{mon}}$  membership problem

---

**Input:** A timed quantitative word

$w = (u_1, \tau_1), (u_2, \tau_2), \dots, (u_n, \tau_n)$  and an LHA  $\mathcal{M} = (L, F, \mathbb{X}, \text{Init}, \mathcal{F}, \text{Inv}, E)$ .

**Output:** The Boolean sequence

$Result_1, \dots, Result_n$ , where

$Result_i = \top \iff w[i] \in \mathcal{L}_{\text{mon}}(\mathcal{M})$

1  $State_0 \leftarrow \{(\ell, v) \mid \ell \in L, v \in \text{Init}(\ell)\}$

2 **for**  $i \leftarrow 1$  **to**  $n$  **do**

3      $State'_i \leftarrow$  reachable states from  $State_{i-1}$  in duration  $\tau_i - \tau_{i-1}$

4      $State_i \leftarrow \{(\ell, v) \in State'_i \mid v = u_i\}$

5      $Result_i \leftarrow \exists(\ell, v) \in State_i. \ell \in F$

---

$\mathcal{L}_{\text{mon}}(\mathcal{M})$ . In line 1, we construct the initial states  $State_0$ . We note that although  $\text{Init}(\ell)$  is, in general, an infinite set, it is given as a convex polyhedron, and we can represent  $State_0$  as a finite list of pairs of a location and a convex polyhedron.

From line 2 to line 5 is the main part of Procedure 1: we incrementally compute  $State_i$  and  $Result_i$ . In line 3, we compute the reachable states  $State'_i$  from  $State_{i-1}$  after the executions of duration  $\tau_i - \tau_{i-1}$ . This part is essentially the same as the bounded-time reachability analysis, and thus, it is undecidable for LHAs in general [Bri+11]. Nevertheless, in practice, the reachable states  $State'_i$  are usually effectively computable as a finite union of convex polyhedra.

In line 4, we require  $State_i$  to be the subset of  $State'_i$  compatible with the current observation  $u_i$ . Thanks to this requirement, we can take into account of the next sample just by the forward reachability analysis. Finally, in line 5, we determine the partial result  $Result_i$  by checking the reachability to the accepting locations.

An example is in the appendix (Example 21).

The intermediate states set  $State_i$  is the set of the last states of the runs of  $\mathcal{M}$  associated with  $w[i]$  and of duration  $\tau_i$ . Therefore, we have the following correctness theorem.

**Theorem 19** (correctness of Procedure 1). *Given a timed quantitative word  $w$  and an LHA  $\mathcal{M}$ , Procedure 1 returns the sequence  $Result_1, \dots, Result_n$  satisfying  $Result_i = \top \iff w[i] \in \mathcal{L}_{\text{mon}}(\mathcal{M})$ .*  $\square$

Table 1: Summary of the benchmarks

Name	Dimension ( $= d$ )	# of locs.	max. length of logs
ACCC	5,10,15	$d + 1$	1,000
ACCI	2	4	100,000
ACCD	2, 3, $\dots$ , 7	$2^d$	1,000
NAV	4	18	150

## 8 Experimental Evaluation

We experimentally evaluated our model-bounded monitoring scheme using the two procedures for  $\mathcal{L}_{\text{mon}}$  membership. For the first procedure via reachability analysis (in Section 6), we used PHAVerLite [BZ19] for conducting reachability analysis. For the second direct procedure (in Section 7), we implemented a prototypical tool HAMONI.

We pose the following research questions.

- RQ1** Is our dedicated implementation HAMONI worthwhile, performance-wise?
- RQ2** Is HAMONI scalable w.r.t. the length of the input log?
- RQ3** How is the scalability of PHAVerLite and HAMONI w.r.t. the dimension of the bounding model?
- RQ4** Is there any relationship between the robustness of the log and the precision of model-bounded monitoring? Moreover, can we reduce the sampling interval in the model-bounded monitoring scheme in Fig. 4 without causing significant false alarms?

### 8.1 Benchmarks

Table 1 summarizes the benchmarks for both scalability and precision experiments.

#### 8.1.1 Benchmarks for RQ1–3

In the scalability experiments to answer RQ1–3, we used the following three benchmarks on adaptive cruise controller: Piecewise-Constant ACC (ACCC); Interval ACC (ACCI); and Diagonal ACC (ACCD). The bounding models for the benchmarks are mostly taken from the literature (see below); they express keeping the intervehicular distance by switching between the normal cruise mode and the recovery mode. This is much like Fig. 6a.

The input logs  $w$  were randomly generated by following the flows and the transitions of the bounding model  $\mathcal{M}$ . This means that our SUM is the bounding model itself (Fig. 4). We note that this coincidence is not mandatory. **Piecewise-Constant ACC (ACCC)** The bounding models for ACCC are taken from [BRS19]. The accepting

locations of ACCC happen to be unreachable, thus there will be no alerts. This is no problem for the scalability evaluation. In ACCC, the velocities of the cars at each location are constant. ACCC contains three LHAs of dimensions 5, 10, and 15. Fig. 17 (in Appendix D) is the LHA of dimension 5.

**Interval ACC (ACCI)** ACCI is a variant of the ACCC benchmark. In the bounding model for ACCI, the velocities of the cars at each location are nondeterministically chosen from the given interval. It is shown in Fig. 6b.

**Diagonal ACC (ACCD)** The bounding models for ACCD are taken from [Fre+19]. In ACCD, the velocities of the cars at each location are constrained by the following diagonal constraints (i. e., constraints of the form  $x_i - x_j \bowtie n, n \in \mathbb{Z}$ ): when recovering the distance between  $x_i$  and  $x_{i+1}$ , we have  $|\dot{x}_i - \dot{x}_{i+1} - \varepsilon| < 1$ , where  $\varepsilon$  is the slow-down parameter; otherwise, we have  $|\dot{x}_i - \dot{x}_{i+1}| < 1$ . We used  $\varepsilon = 0.9$  and  $\varepsilon = 2.0$ . The safety specification in ACCD is  $x_i > x_{i-1}$  for each  $i$ . ACCD contains six LHAs of dimensions from 2 to 7. The LHAs of dimension 2 are shown in Fig. 14 (in Appendix D).

#### 8.1.2 Benchmark for RQ4

In the precision experiments to answer RQ4, we used the robot navigation benchmark (NAV). The original system model is an *affine* hybrid automaton [DHR05] and the bounding model is constructed by the *projection* overapproximation in [Fre08, Section 3.2]. We use Definition 12 to construct  $\mathcal{M}_{-\varphi}$ . The input logs  $w$  were generated by a simulation of a Simulink implementation of the original system model.

The original model for NAV is taken from [FI04]. To obtain an LHA  $\mathcal{M}$ , we used the projection overapproximation in [Fre08] with an additional invariant  $v_x \in [-1, 1], v_y \in [-1, 1]$ . We constructed the bounding model  $\mathcal{M}_{-\varphi}$  by the construction in Definition 12, where the monitored safety specification  $\varphi$  is  $v_x \in [-1, 1] \wedge v_y \in [-1, 1]$ . We note that the bounding model  $\mathcal{M}_{-\varphi}$  does not contain the invariant  $v_x \in [-1, 1], v_y \in [-1, 1]$ , and the accepting locations for the safety violation are reachable.

## 8.2 Experiments

### 8.2.1 Scalability experiments

For the reachability analysis in the procedure presented in Section 6, we used PHAVerLite 0.2.1. PHAVerLite relies on PPLite [BZ19] to compute symbolic states. We

Table 2: Experiment result on ACCC [sec.]

dim.	len.	PHAVerLite	HAMONI	dim.	len.	PHAVerLite	HAMONI
5	10	500.02	0.46	5	500	480.41	2.27
5	25	540.02	0.86	5	600	500.40	2.12
5	50	480.06	0.78	5	700	520.40	2.37
5	75	580.01	0.79	5	800	540.33	2.58
5	100	520.07	1.21	5	900	580.12	2.76
5	200	560.06	1.00	5	1000	560.26	3.26
5	300	540.13	1.37	10	10	267.60	204.79
5	400	T.O.	1.77	15	10	T.O.	T.O.

Table 3: Experiment result on ACCI [sec.]

len.	PHAVerLite	HAMONI	len.	PHAVerLite	HAMONI
10000	13.42	2.18	60000	116.67	12.86
20000	40.29	4.29	70000	26.60	15.00
30000	83.39	6.43	80000	227.29	17.14
40000	141.55	8.56	90000	259.17	19.28
50000	225.23	10.76	100000	227.12	21.45

implemented an OCaml program and a Python script to construct a PHAVerLite model from an LHA  $\mathcal{M}$  and a timed quantitative word  $w$ . For the procedure presented in Section 7, we implemented HAMONI in C++ with Parma Polyhedra Library (PPL) [BHZ08] and compiled using GCC 7.4.0. In both PHAVerLite and HAMONI, closed convex polyhedra are used to analyze the reachability [BZ19].

Since the difficulty of the  $\mathcal{L}_{\text{mon}}$  membership problem depends on the given timed quantitative word  $w$ , we randomly generated 30 logs for each experiment setting and measured the average of the execution time. The sampling interval of  $w$  is from 1 to 5 seconds, uniformly distributed. The timeout is 10 minutes. We conducted the experiments on an Amazon EC2 c4.large instance (2.9 GHz Intel Xeon E5-2666 v3, 2 vCPUs, and 3.75 GiB RAM) that runs Ubuntu 18.04 LTS (64 bit). Tables 2 to 4 summarize the experiment results.

**RQ1: worthwhileness of a dedicated implementation** In Tables 2 to 4, we observe that HAMONI tends to outperform PHAVerLite. Especially, in Table 2, we observe that for ACCC, HAMONI performed drastically faster than PHAVerLite for dimension 5. This is because PHAVerLite is not a specific tool for the  $\mathcal{L}_{\text{mon}}$  membership problem but a tool for reachability analysis in general. Thus, despite the engineering cost for the implementation that is not small, a dedicated solver i.e., HAMONI, is worthwhile.

However, in Table 4, we also observe that in ACCD, when the dimension of the LHA is relatively large and the timed quantitative word is not too long, PHAVerLite often outperformed HAMONI. This is because of the optimized reachability analysis algorithm in PHAVerLite. Optimization of the reachability analysis algorithm in HA-

Table 4: Experiment result on ACCD [sec.]

dim.	len.	$\varepsilon = 0.9$		$\varepsilon = 2.0$	
		PHAVerLite	HAMONI	PHAVerLite	HAMONI
2	25	0.03	0.00	0.03	0.00
2	50	0.04	0.00	0.04	0.00
2	75	0.05	0.01	0.05	0.00
2	100	0.06	0.01	0.06	0.01
2	200	0.12	0.02	0.11	0.01
2	300	0.16	0.02	0.16	0.02
2	400	0.21	0.03	0.20	0.02
2	500	0.27	0.04	0.26	0.03
2	600	0.32	0.04	0.30	0.03
2	700	0.35	0.05	0.36	0.04
2	800	0.40	0.05	0.40	0.05
2	900	0.46	0.06	0.45	0.05
2	1000	0.51	0.07	0.50	0.06
3	10	0.05	0.02	0.04	0.01
3	25	0.08	0.03	0.08	0.02
3	50	0.14	0.04	0.12	0.03
3	75	0.19	0.05	0.18	0.03
3	100	0.23	0.05	0.22	0.04
3	200	0.44	0.07	0.42	0.05
3	300	0.65	0.09	0.62	0.06
3	400	0.84	0.12	0.82	0.09
3	500	1.07	0.15	1.00	0.10
3	600	1.25	0.16	1.24	0.11
3	700	1.46	0.16	1.42	0.13
3	800	1.73	0.20	1.61	0.13
3	900	1.84	0.19	1.83	0.16
3	1000	2.04	0.20	2.00	0.17
4	10	0.28	0.35	0.22	0.42
4	25	0.46	0.46	0.34	0.22
4	50	0.66	0.42	0.59	0.38
4	75	0.92	0.47	0.82	0.36
4	100	1.21	0.64	1.05	0.35
4	200	2.13	0.82	2.02	0.87
4	300	2.98	0.74	2.81	0.40
4	400	3.98	0.92	3.76	0.62
4	500	4.79	0.85	4.69	0.65
4	600	5.71	0.86	5.63	0.68
4	700	6.60	0.81	6.53	0.93
4	800	7.67	1.15	7.28	0.97
4	900	8.39	1.04	0.02	0.87
4	1000	9.26	1.18	9.15	0.89
5	10	2.65	7.51	22.38	29.33
5	25	3.84	10.13	3.05	9.22
5	50	5.25	9.68	12.93	31.06
5	75	7.13	14.41	18.32	29.93
5	100	8.14	12.87	7.58	21.13
5	200	11.39	8.94	29.52	36.17
5	300	17.33	15.03	15.38	14.56
5	400	20.44	9.81	19.47	15.34
5	500	24.87	11.72	36.83	34.12
5	600	28.78	8.79	28.47	16.37
5	700	35.14	17.05	32.86	16.10
5	800	37.76	12.41	41.97	39.07
5	900	42.22	9.83	48.26	46.68
5	1000	47.11	11.55	57.49	54.44
6	10	47.42	221.55	80.26	428.66
6	25	41.18	165.63	76.40	510.55
6	50	65.36	243.97	120.77	518.01
6	75	58.21	173.46	125.61	480.19
6	100	87.47	209.53	131.88	421.59
7	10	525.07	560.69	T.O.	594.05
7	25	489.35	559.56	T.O.	526.14
7	50	514.10	562.68	T.O.	566.01
7	75	T.O.	588.23	T.O.	583.49
7	100	T.O.	577.29	T.O.	578.39

MONI e.g., utilizing the techniques in [BZ19; CÁF11], is a future work.

**RQ2: scalability w.r.t. the word length** Figs. 9 and 10 show the execution time of HAMONI w.r.t. the length of the timed quantitative word for selected experiment settings. See Appendix C for the other settings.

In Figs. 9 and 10a, we observe that when the dimension of the LHA is not large, the execution time was more or less linear to the word length. This is because, when the number of the intermediate states ( $State_i$  in Procedure 1) is constant, the execution time of the bounded-time reachability analysis (line 3 of Procedure 1) is constant for each iteration, and the execution time of Procedure 1 is linear to the word length. Thanks to the merging of the convex polyhedra, such saturation often happens when the word length is long enough for the complexity of the LHA.

In Fig. 10b, we observe that for ACCI of dimension 5, the execution time was more or less constant w.r.t. the word length. Such behavior also happens for other benchmarks when the word length is short e.g., when the word length is less than 200 for ACCC of dimension 5.

Overall, in our experiments, the execution time was at most linear, and we conclude that at least for many benchmarks, HAMONI is scalable to the word length.

**RQ3: scalability w.r.t. the dimensionality of the LHA** Fig. 11 shows the execution time of PHAVerLite and HAMONI to the dimension of the LHA, where the length of the log is fixed to be 100. As we mentioned in Section 8.2.1, the sampling interval is from 1 to 5 seconds, uniformly distributed; thus, each  $w$  spans 300 seconds on average. Note that the  $y$ -axis of Fig. 11 follows a logarithmic scale.

In Fig. 11, we observe that the execution time is more or less exponential to the dimension of the LHA. This is due to the exponential complexity of the convex polyhedra operations. However, in Tables 2 to 4, we observe that for any benchmark (excluding ACCI, which consists of only one LHA and is not suitable for this discussion), HAMONI can effectively process a huge log up to around 5 dimensions. This is important for monitoring where the log tends to be huge while the dimension of the bounding model may not be much large.

In contrast, in Fig. 11, we observe that PHAVerLite is more scalable to the model dimension than HAMONI. This is thanks to the optimized convex polyhedra algorithms and implementations in PHAVerLite. Again, our future work will consist in optimizing HAMONI e.g., by using the techniques from [BZ19; CÁF11].

Overall, our experiment results suggest that for signals of up to 5 dimensions, our monitoring works more or less

in real-time because both implementations handle the logs spanning at least 100 seconds (a few paragraphs ago) in less than 30 seconds on average (Fig. 11).

## 8.2.2 Precision Experiments

In the precision experiments, since the result of the procedures presented in Sections 6 and 7 are the same, we only used HAMONI to evaluate the precision of model-bounded monitoring. Since NAV has a nondeterminism in its initial configuration, we randomly generated 100 logs for NAV. From a dense enough raw log, we generated timed quantitative words  $w$  with constant sampling interval. We tried the following 10 sampling intervals: 100, 200, 300,  $\dots$ , 1000 milliseconds.

**RQ4: robustness vs. precision** Fig. 12 shows the shortest sampling interval with no false alarms and the robustness for each log of NAV. Since the monitored specification is  $v_x \in [-1, 1] \wedge v_y \in [-1, 1]$ , the robustness i.e., the satisfaction degree of the monitored specification, is the minimum value of  $\min\{1 - |v_x|, 1 - |v_y|\}$  in the whole execution in the raw log.

In Fig. 12, we observe that when the robustness is small i.e., the execution is close to violation, model-bounded monitoring tends to cause false alarms with a short sampling interval. This is because when the sampled log is coarse, the overapproximation in the model-bounded monitoring is rough, and near unsafe behavior is deemed to be unsafe. This observation suggests the following process to decide a reasonable sampling interval:

1. assume false alarms with small robustness (e.g., robustness  $\leq 0.15$ ) are acceptable;
2. we plot the relationship between the robustness and the shortest sampling interval without false alarms e.g., Fig. 12; and
3. we decide the sampling interval so that the amount of the false alarms with high robustness becomes reasonably small, e.g., in Fig. 12, if the false alarms with robustness  $\leq 0.15$  are acceptable, 600 ms is a reasonable sampling interval.

We note that our LHA construction does not utilize any runtime information, and the precision of the approximation is limited. It is a future work to reduce the false alarms by improving the approximation utilizing the runtime information by an online construction of the LHAs e.g., the iterative refinement in [Fre08, Section 3.2].

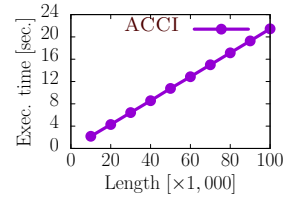
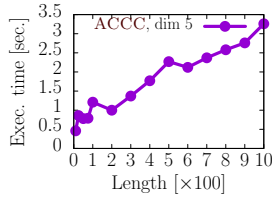
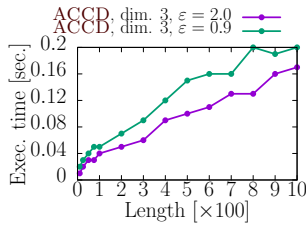
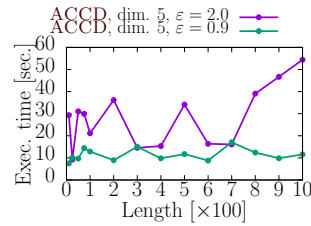


Figure 9: Execution time of HAMONI for ACCC dimension 5 (left) and ACCI (right)



(a) ACCD of dimension 3



(b) ACCD of dimension 5

Figure 10: Execution time of HAMONI for ACCD

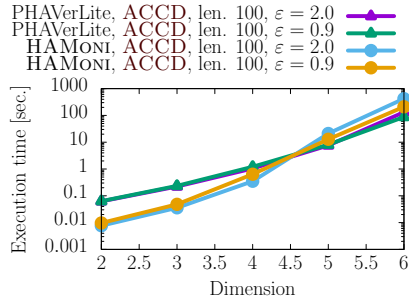


Figure 11: Execution time of PHAVerLite and HAMONI for ACCD fixing the word length to be 100

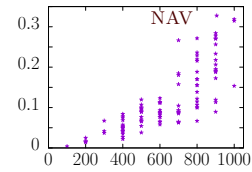


Figure 12: Robustness (vertical) vs. shortest sampling interval with no false alarms [ms.] (horizontal)

## 9 Conclusions and Perspectives [AKV98]

Based on a novel language notion  $\mathcal{L}_{\text{mon}}$  for LHAs, we formulated what we call the *model-bounded monitoring* scheme for hybrid systems. It features the use of a bounding model of the system to bridge the gap between (continuous-time) system behaviors and (discrete-time) logs that a monitor can access. While the  $\mathcal{L}_{\text{mon}}$  membership problem is undecidable, our two partial algorithms (especially our dedicated HAMONI) work well for automotive platooning benchmarks. Overall, our results show the power of symbolic manipulation of polyhedra in the domain of cyber-physical systems.

So far, HAMONI is a quickly developed prototype. Optimization of the reachability analysis with the technique in [BZ19; CÁF11] is a first future work. In addition, importing the latest optimizations from PHAVerLite [BZ19] into HAMONI is on our agenda. Further evaluation of the precision of model-bounded monitoring using a real-world industry example is also a future work.

One potential extension of the  $\mathcal{L}_{\text{mon}}$  membership problem is to make it more quantitative: returning a *distance* between the monitored language  $\mathcal{L}_{\text{mon}}(\mathcal{M})$  and the observation  $w$  rather than checking the membership of the observation  $w$  to the monitored language  $\mathcal{L}_{\text{mon}}(\mathcal{M})$ . This makes the result of the analysis even more useful. Another extension is to use intervals rather than points for the measured values to cope with the uncertainty in the measured values.

A testing-based approach, e.g., in [DN09], instead of the current *guaranteed* approach by the convex polyhedra analysis is a future work for more efficiency at the expense of the safety guarantee.

Being parametric can be extremely useful when performing monitoring or in the close area of parameter identification [Asa+11; BFM18; WAH19]. Parametric monitoring consists in exhibiting parameter valuations for which a specification is violated (or correct). Parametrization of the current framework is future work.

## References

- [AHW18] Étienne André, Ichiro Hasuo, and Masaki Waga. “Offline timed pattern matching under uncertainty”. In: *ICECCS* (Dec. 12–14, 2018). Ed. by Anthony Widjaja Lin and Jun Sun. Melbourne, Australia: IEEE Computer Society, 2018, pp. 10–20. DOI: [10.1109/ICECCS2018.2018.00010](https://doi.org/10.1109/ICECCS2018.2018.00010) (cit. on pp. 6, 10).
- [Asa+11] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. “Parametric Identification of Temporal Properties”. In: *RV* (Sept. 27–30, 2011). Ed. by Sarfraz Khurshid and Koushik Sen. Vol. 7186. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer, 2011, pp. 147–160. DOI: [10.1007/978-3-642-29860-8\\_12](https://doi.org/10.1007/978-3-642-29860-8_12) (cit. on p. 16).
- [Bak+18] Alexey Bakhirkin, Thomas Ferrère, Dejan Nickovic, Oded Maler, and Eugene Asarin. “Online Timed Pattern Matching Using Automata”. In: *FORMATS* (Sept. 4–6, 2018). Ed. by David N. Jansen and Prabhakar Pavithra. Vol. 11022. Lecture Notes in Computer Science. Beijing, China: Springer, 2018, pp. 215–232. DOI: [10.1007/978-3-030-00151-3\\_13](https://doi.org/10.1007/978-3-030-00151-3_13) (cit. on p. 6).
- [BFM18] Alexey Bakhirkin, Thomas Ferrère, and Oded Maler. “Efficient Parametric Identification for STL”. In: *HSCC* (Apr. 11–13, 2018). Porto, Portugal: ACM, 2018, pp. 177–186. DOI: [10.1145/3178126.3178132](https://doi.org/10.1145/3178126.3178132) (cit. on p. 16).
- [BGF18] Reza Babaei, Arie Gurfinkel, and Sebastian Fischmeister. “Prevent: A Predictive Run-Time Verification Framework Using Statistical Learning”. In: *SEFM* (June 27–29, 2018). Ed. by Einar Broch Johnsen and Ina Schaefer. Vol. 10886. Lecture Notes in Computer Science. Toulouse, France: Springer, 2018, pp. 205–220. ISBN: 978-3-319-92969-9. DOI: [10.1007/978-3-319-92970-5\\_13](https://doi.org/10.1007/978-3-319-92970-5_13) (cit. on p. 6).
- [BHZ08] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. “The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems”. In: *Science of Computer Programming* 72.1-2 (2008), pp. 3–21. DOI: [10.1016/j.scico.2007.08.001](https://doi.org/10.1016/j.scico.2007.08.001) (cit. on p. 13).
- [Bri+11] Thomas Brihaye, Laurent Doyen, Gilles Geeraerts, Joël Ouaknine, Jean-François Raskin, and James Worrell. “On Reachability for Hybrid Automata over Bounded Time”. In: *ICALP Part II* (July 4–8, 2011). Ed. by Luca Aceto, Monika Henzinger, and Jiri Sgall. Vol. 6756. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, 2011, pp. 416–427. DOI: [10.1007/978-3-642-22012-8\\_33](https://doi.org/10.1007/978-3-642-22012-8_33) (cit. on pp. 11, 19).



- [BRS19] Lei Bu, Rajarshi Ray, and Stefan Schupp. “ARCH-COMP19 Category Report: Bounded Model Checking of Hybrid Systems with Piecewise Constant Dynamics”. In: *ARCH@CPSIoTWeek* (Apr. 15, 2019). Vol. 61. EPiC Series in Computing. Montréal, QC, Canada: EasyChair, 2019, pp. 120–128 (cit. on p. 12).
- [Bu+19] Lei Bu, Jiawan Wang, Yuming Wu, and Xuan-dong Li. “From Bounded Reachability Analysis of Linear Hybrid Automata to Verification of Industrial CPS and IoT”. In: *SETSS* (Apr. 21–27, 2019). Ed. by Jonathan P. Bowen, Zhiming Liu, and Zili Zhang. Vol. 12154. Lecture Notes in Computer Science. Chongqing, China: Springer, 2019, pp. 10–43. DOI: [10.1007/978-3-030-55089-9\\_2](https://doi.org/10.1007/978-3-030-55089-9_2) (cit. on p. 9).
- [BZ19] Anna Becchi and Enea Zaffanella. “Revisiting Polyhedral Analysis for Hybrid Systems”. In: *SAS* (Oct. 8–11, 2019). Ed. by Bor-Yuh Evan Chang. Vol. 11822. Lecture Notes in Computer Science. Porto, Portugal: Springer, 2019, pp. 183–202. DOI: [10.1007/978-3-030-32304-2\\_10](https://doi.org/10.1007/978-3-030-32304-2_10) (cit. on pp. 5, 10, 12–14, 16).
- [CÁF11] Xin Chen, Erika Ábrahám, and Goran Frehse. “Efficient Bounded Reachability Computation for Rectangular Automata”. In: *RP* (Sept. 28–30, 2011). Ed. by Giorgio Delzanno and Igor Potapov. Vol. 6945. Lecture Notes in Computer Science. Genoa, Italy: Springer, 2011, pp. 139–152. DOI: [10.1007/978-3-642-24288-5\\_13](https://doi.org/10.1007/978-3-642-24288-5_13) (cit. on pp. 14, 16).
- [DHR05] Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. “Automatic Rectangular Refinement of Affine Hybrid Systems”. In: *FORMATS* (Sept. 26–28, 2005). Ed. by Paul Pettersson and Wang Yi. Vol. 3829. Lecture Notes in Computer Science. Uppsala, Sweden: Springer, 2005, pp. 144–161. DOI: [10.1007/11603009\\_13](https://doi.org/10.1007/11603009_13) (cit. on pp. 12, 20).
- [DN09] Thao Dang and Tarik Nahhal. “Coverage-guided test generation for continuous and hybrid systems”. In: *Formal Methods in System Design* 34.2 (2009), pp. 183–213. DOI: [10.1007/s10703-009-0066-0](https://doi.org/10.1007/s10703-009-0066-0) (cit. on p. 16).
- [FI04] Ansgar Fehnker and Franjo Ivancic. “Benchmarks for Hybrid Systems Verification”. In: *HSCC* (Mar. 25–27, 2004). Ed. by Rajeev Alur and George J. Pappas. Vol. 2993. Lecture Notes in Computer Science. Philadelphia, PA, USA: Springer, 2004, pp. 326–341. DOI: [10.1007/978-3-540-24743-2\\_22](https://doi.org/10.1007/978-3-540-24743-2_22) (cit. on p. 12).
- [FP09] Georgios E. Fainekos and George J. Pappas. “Robustness of temporal logic specifications for continuous-time signals”. In: *Theoretical Computer Science* 410.42 (2009), pp. 4262–4291. DOI: [10.1016/j.tcs.2009.06.021](https://doi.org/10.1016/j.tcs.2009.06.021) (cit. on p. 6).
- [Fre+19] Goran Frehse, Alessandro Abate, Dieky Adzkiya, Anna Becchi, Lei Bu, Alessandro Cimatti, Mirco Giacobbe, Alberto Griggio, Sergio Mover, Muhammad Syifa’ul Mufid, Idriss Riouak, Stefano Tonetta, and Enea Zaffanella. “ARCH-COMP19 Category Report: Hybrid Systems with Piecewise Constant Dynamics”. In: *ARCH@CPSIoTWeek* (Apr. 15, 2019). Vol. 61. EPiC Series in Computing. Montréal, QC, Canada: EasyChair, 2019, pp. 1–13 (cit. on p. 12).
- [Fre08] Goran Frehse. “PHAVER: Algorithmic Verification of Hybrid Systems Past HyTech”. In: *International Journal on Software Tools for Technology Transfer* 10.3 (May 2008), pp. 263–279. ISSN: 1433-2779. DOI: [10.1007/s10009-007-0062-x](https://doi.org/10.1007/s10009-007-0062-x) (cit. on pp. 5, 12, 14, 20).
- [Gub+13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. “Internet of Things (IoT): A vision, architectural elements, and future directions”. In: *Future Generation Computing Systems* 29.7 (2013), pp. 1645–1660. DOI: [10.1016/j.future.2013.01.010](https://doi.org/10.1016/j.future.2013.01.010) (cit. on p. 6).
- [HMP92] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. “Timed Transition Systems”. In: *REX* (June 3–7, 1991). Ed. by J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg. Vol. 600. Lecture Notes in Computer Science. Mook, The Netherlands: Springer, 1992, pp. 226–251. DOI: [10.1007/BFb0031995](https://doi.org/10.1007/BFb0031995) (cit. on p. 7).
- [HPR94] Nicolas Halbwachs, Yann-Éric Proy, and Pascal Raymond. “Verification of Linear Hybrid Systems by Means of Convex Approximations”. In: *SAS* (Sept. 28–30, 1994). Ed. by Baudouin Le Charlier. Vol. 864. Lecture Notes in Computer Science. Namur, Belgium: Springer, 1994, pp. 223–237. DOI: [10.1007/3-540-58485-4\\_43](https://doi.org/10.1007/3-540-58485-4_43) (cit. on pp. 3, 7).
- [MN04] Oded Maler and Dejan Nickovic. “Monitoring Temporal Properties of Continuous Signals”. In: *FORMATS and FTRTFT* (Sept. 22–24, 2004). Ed. by Yassine Lakhnech and Sergio Yovine. Vol. 3253. Lecture Notes in Computer Science. Grenoble, France: Springer, 2004, pp. 152–166. DOI: [10.1007/978-3-540-30206-3\\_12](https://doi.org/10.1007/978-3-540-30206-3_12) (cit. on p. 6).

- [Pin+17] Srinivas Pinisetty, Thierry Jéron, Stavros Tripakis, Yliès Falcone, Hervé Marchand, and Viorel Preoteasa. “Predictive runtime verification of timed properties”. In: *Journal of Systems and Software* 132 (2017), pp. 353–365. DOI: [10.1016/j.jss.2017.06.060](https://doi.org/10.1016/j.jss.2017.06.060) (cit. on p. 6).
- [QD20] Xin Qin and Jyotirmoy V. Deshmukh. “Clairvoyant Monitoring for Signal Temporal Logic”. In: *FORMATS* (Sept. 1–3, 2020). Ed. by Nathalie Bertrand and Nils Jansen. Vol. 12288. Lecture Notes in Computer Science. Vienna, Austria: Springer, 2020, pp. 178–195. DOI: [10.1007/978-3-030-57628-8\\_11](https://doi.org/10.1007/978-3-030-57628-8_11) (cit. on p. 6).
- [Ras05] Jean-François Raskin. “An Introduction to Hybrid Automata”. In: *Handbook of Networked and Embedded Control Systems*. Ed. by Dimitrios Hristu-Varsakelis and William S. Levine. Birkhäuser, 2005, pp. 491–518 (cit. on p. 7).
- [Ulu+14] Dogan Ulus, Thomas Ferrère, Eugene Asarin, and Oded Maler. “Timed Pattern Matching”. In: *FORMATS* (Sept. 8–10, 2014). Ed. by Axel Legay and Marius Bozga. Vol. 8711. Lecture Notes in Computer Science. Florence, Italy: Springer, 2014, pp. 222–236. DOI: [10.1007/978-3-319-10512-3\\_16](https://doi.org/10.1007/978-3-319-10512-3_16) (cit. on p. 6).
- [UM18] Dogan Ulus and Oded Maler. “Specifying Timed Patterns using Temporal Logic”. In: *HSCC* (Apr. 11–13, 2018). Porto, Portugal: ACM, 2018, pp. 167–176. DOI: [10.1145/3178126.3178129](https://doi.org/10.1145/3178126.3178129) (cit. on p. 6).
- [WA19] Masaki Waga and Étienne André. “Online Parametric Timed Pattern Matching with Automata-Based Skipping”. In: *NFM* (May 7–9, 2019). Ed. by Julia Badger and Kristin Yvonne Rozier. Vol. 11460. Lecture Notes in Computer Science. Houston, TX, USA: Springer, 2019, pp. 371–389. DOI: [10.1007/978-3-030-20652-9\\_26](https://doi.org/10.1007/978-3-030-20652-9_26) (cit. on p. 6).
- [Wag19] Masaki Waga. “Online Quantitative Timed Pattern Matching with Semiring-Valued Weighted Automata”. In: *FORMATS* (Aug. 27–29, 2019). Ed. by Étienne André and Mariëlle Stoelinga. Vol. 11750. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, 2019, pp. 3–22. DOI: [10.1007/978-3-030-29662-9\\_1](https://doi.org/10.1007/978-3-030-29662-9_1) (cit. on p. 6).
- [WAH19] Masaki Waga, Étienne André, and Ichiro Hasuo. “Symbolic Monitoring against Specifications Parametric in Time and Data”. In: *CAV, Part I* (July 15–18, 2019). Ed. by Işıl Dillig and Serdar Tasiran. Vol. 11561. Lecture Notes in Computer Science. New York City, USA: Springer, 2019, pp. 520–539. DOI: [10.1007/978-3-030-25540-4\\_30](https://doi.org/10.1007/978-3-030-25540-4_30) (cit. on pp. 6, 16).
- [WHS17] Masaki Waga, Ichiro Hasuo, and Kohei Sue-naga. “Efficient Online Timed Pattern Matching by Automata-Based Skipping”. In: *FORMATS* (Sept. 5–7, 2019). Ed. by Alessandro Abate and Gilles Geeraerts. Vol. 10419. Lecture Notes in Computer Science. Berlin, Germany: Springer, 2017, pp. 224–243. DOI: [10.1007/978-3-319-65765-3\\_13](https://doi.org/10.1007/978-3-319-65765-3_13) (cit. on p. 6).
- [ZLD12] Xian Zhang, Martin Leucker, and Wei Dong. “Runtime Verification with Predictive Semantics”. In: *NFM* (Apr. 3–5, 2012). Ed. by Alwyn Goodloe and Suzette Person. Vol. 7226. Lecture Notes in Computer Science. Norfolk, VA, USA: Springer, 2012, pp. 418–432. ISBN: 978-3-642-28890-6. DOI: [10.1007/978-3-642-28891-3\\_37](https://doi.org/10.1007/978-3-642-28891-3_37) (cit. on p. 6).

## A Omitted proofs

### A.1 Proof of Theorem 17

The proof of Theorem 17 is as follows.

*Proof.* We prove the claim by a reduction from the bounded-time reachability of the LHA  $\mathcal{M} = (L, F, \mathbb{X}, \text{Init}, \mathcal{F}, \text{Inv}, E)$ . Let  $\tau \in \mathbb{R}_{>0}$  be the time bound of the reachability checking. Let  $u: \mathbb{X} \rightarrow \mathbb{R}$  be the valuation satisfying  $u(x) = 0$  for any  $x \in \mathbb{X}$ ,  $w = (u, \tau)$ , and  $\mathcal{M}' = (L \sqcup \{\ell_f\}, \{\ell_f\}, \mathbb{X} \sqcup \{t\}, \text{Init}, \mathcal{F}', \text{Inv}', E \sqcup E')$ , where:

- $\mathcal{F}'$  is  $\mathcal{F}'(\ell) = \{\mathcal{F}(\ell) \wedge t = 1\}$  for any  $\ell \in L$ , and  $\mathcal{F}'(\ell_f)$  is such that  $\dot{x} = 0$  for any  $x \in \mathbb{X}$  and  $t = 1$ ;
- $\text{Inv}'$  is  $\text{Inv}'(\ell) = \text{Inv}(\ell)$  for  $\ell \in L$  and  $\text{Inv}'(\ell_f) = \top$ ; and
- $E' = \{(\ell, \{t \leq \tau\}, \mu, \ell_f) \mid \ell \in F, \mu(x) = 0 \text{ for any } x \in \mathbb{X}\}$ .

We have  $(u, \tau) \in \mathcal{L}_{\text{mon}}(\mathcal{M}')$  if and only if  $F$  is reachable in  $\mathcal{M}$  within  $\tau$ . Since we have  $|w| = 1$ , the bounded-time reachability of the LHA  $\mathcal{M}$ , which is undecidable [Bri+11], is reduced to the emptiness checking of  $\mathcal{C}(w, \mathcal{M}')$ .  $\square$

### A.2 Proof of Theorem 19

First, we prove the following lemma.

**Lemma 20.** *Let  $w$  be a timed quantitative word  $w = (u_1, \tau_1), \dots, (u_m, \tau_m)$ ,  $\mathcal{M}$  be an LHA  $\mathcal{M} = (L, F, \mathbb{X}, \text{Init}, \mathcal{F}, \text{Inv}, E)$ , and  $\text{State}_i \in L \times (\mathbb{R}_{\geq 0})^{\mathbb{X}}$  be the  $\text{State}_i$  in line 4 of Procedure 1, where  $i \in \{1, 2, \dots, m\}$ . For any  $i \in \{1, 2, \dots, m\}$ ,  $\ell \in L$ , and  $v \in (\mathbb{R}_{\geq 0})^{\mathbb{X}}$ , we have  $(\ell, v) \in \text{State}_i$  if and only if there is a run  $\rho_i = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell, v)$  of  $\mathcal{M}$  associated with  $w[i]$  and satisfying  $\text{Dur}(\rho_i) = \tau_i$ .*

*Proof.* We prove Lemma 20 by induction on  $i$ . When  $i = 1$ ,  $\text{State}_1$  is the set of states reachable from  $\{(\ell_0, v_0) \mid \ell_0 \in L, v_0 \in \text{Init}(\ell)\}$  in  $\tau_1$  and satisfying  $v = u_1$ . Therefore, for any  $(\ell_1, v_1) \in \text{State}_1$ , there are  $\ell_0 \in L, v_0 \in \text{Init}(\ell)$ , and a run  $\rho_1 = (\ell_0, v_0), \rightarrow_1, s_1, \dots, \rightarrow_n, (\ell_1, v_1)$  of  $\mathcal{M}$  satisfying  $\text{Dur}(\rho_1) = \tau_1$ . By Definition 10, such  $\rho_1$  is associated to  $w[1]$ , and thus, when we have  $(\ell_1, v_1) \in \text{State}_1$ , there is a run  $\rho_1 = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell_1, v_1)$  of  $\mathcal{M}$  associated with  $w[1]$  and satisfying  $\text{Dur}(\rho_1) = \tau_1$ . When there is a run  $\rho_1 = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell_1, v_1)$  of  $\mathcal{M}$  associated with  $w[1]$  and satisfying  $\text{Dur}(\rho_1) = \tau_1$ , since  $s_0 \in \text{State}_0$  holds, we have  $(\ell_1, v_1) \in \text{State}_1$ .

When  $i > 1$ , by definition in Procedure 1,  $\text{State}_i$  is the set of states  $(\ell_i, v_i) \in L \times (\mathbb{R}_{\geq 0})^{\mathbb{X}}$  satisfying the following

- There is a state  $(\ell_{i-1}, v_{i-1}) \in \text{State}_{i-1}$  such that  $(\ell_i, v_i)$  is reachable from  $(\ell_{i-1}, v_{i-1})$  in  $\tau_i - \tau_{i-1}$ .
- We have  $v_i = u_i$ .

By definition of the reachability in  $\mathcal{M}$ ,  $\text{State}_i$  is the set of states  $(\ell_i, v_i)$  such that there is an alternating sequence  $\tilde{\rho}_i = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell_i, v_i)$  of concrete states  $s_i \in S$  and transitions  $\rightarrow_i \in \rightarrow$  satisfying the following.

- $s_0 \rightarrow_1 s_1 \cdots \rightarrow_n s_n$
- $s_0 \in \text{State}_{i-1}$
- Sum of the delays in  $\tilde{\rho}_i$  is  $\tau_i - \tau_{i-1}$
- $v_i = u_i$

By induction hypothesis, for any state  $(\ell_{i-1}, v_{i-1}) \in \text{State}_{i-1}$ , there is a run  $\rho_{i-1} = s'_0, \rightarrow'_1, s'_1, \dots, \rightarrow'_n, s'_n = (\ell_{i-1}, v_{i-1})$  of  $\mathcal{M}$  associated with  $w[i-1]$  and satisfying  $\text{Dur}(\rho_{i-1}) = \tau_{i-1}$ . Therefore, for any  $\ell \in L$  and  $v \in (\mathbb{R}_{\geq 0})^{\mathbb{X}}$  satisfying  $(\ell, v) \in \text{State}_i$ , there is a run

$$\rho_i = \rho_{i-1} \cdot \tilde{\rho}_i = s'_0, \rightarrow'_1, s'_1, \dots, \rightarrow'_n, s'_n, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell_i, v_i)$$

of  $\mathcal{M}$ . Moreover, since  $\text{Dur}(\rho_{i-1}) = \tau_{i-1}$  holds,  $\rho_{i-1}$  is associated with  $w[i-1]$ , and the sum of the delays in  $\tilde{\rho}_i$  is  $\tau_i - \tau_{i-1}$ , such  $\rho_i$  satisfies  $\text{Dur}(\rho_i) = \tau_i$  and  $\rho_i$  is associated with  $w[i]$ .

For any run  $\rho_i = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell_i, v_i)$  of  $\mathcal{M}$  associated with  $w[i]$  and satisfying  $\text{Dur}(\rho_i) = \tau_i$ , there is a run  $\rho_{i-1} = s_0, \rightarrow_1, s_1, \dots, \rightarrow_k, s_k = (\ell_{i-1}, v_{i-1})$  of  $\mathcal{M}$  associated with  $w[i-1]$ , and an alternating sequence  $\tilde{\rho}_i = s_k, \rightarrow_{k+1}, s_{k+1}, \dots, \rightarrow_n, s_n$  satisfying the following.

- $\text{Dur}(\rho_{i-1}) = \tau_{i-1}$
- sum of the delays in  $\tilde{\rho}_i$  is  $\tau_i - \tau_{i-1}$

By induction hypothesis, we have  $(\ell_{i-1}, v_{i-1}) \in \text{State}_{i-1}$ . Moreover, by the existence of  $\tilde{\rho}_i$ ,  $(\ell_i, v_i)$  is reachable from  $(\ell_{i-1}, v_{i-1}) \in \text{State}_{i-1}$ , and thus, we have  $(\ell_i, v_i) \in \text{State}_i$ .  $\square$

*Proof of Theorem 19.* For each  $i \in \{1, 2, \dots, m\}$ , by line 5 of Procedure 1, we have  $\text{Result}_i = \top$  if and only if there is  $(\ell, v) \in \text{State}_i$  satisfying  $\ell \in F$ . By Lemma 20, for each  $(\ell, v) \in \text{State}_i$ , there is a run  $\rho_i = s_0, \rightarrow_1, s_1, \dots, \rightarrow_n, s_n = (\ell, v)$  of  $\mathcal{M}$  associated with  $w[i]$ . Therefore, we have  $\text{Result}_i = \top$  if and only if there is an accepting run  $\rho_i$  of  $\mathcal{M}$  associated with  $w[i]$ , which is equivalent to  $w[i] \in \mathcal{L}_{\text{mon}}(\mathcal{M})$ .  $\square$

## B Additional experiment

We also conducted a precision experiment with an additional benchmark `GASBURNER`.

### Shared Gas-Burner (GasBurner) Benchmark

The original model for `GASBURNER` is taken from [DHR05]. In order to obtain an LHA  $\mathcal{M}$ , we used the projection overapproximation in [Fre08]. We constructed the bounding model  $\mathcal{M}_{-\varphi}$  by the construction in Definition 12, where the monitored safety specification  $\varphi$  is  $x_1 \in [0, 100.1) \wedge x_2 \in [0, 100.1)$ . We note that all the upper bounds 100 of the invariants in bounding model  $\mathcal{M}_{-\varphi}$  are replaced with 150, and thus, the accepting locations for the safety violation are reachable. **Experiment result** For `GASBURNER`, the shortest sampling interval without any false alarms is 600 ms and the robustness is 0. This is because of the initial value  $x_1 = 0$ .

In `GASBURNER`, even though the robustness is 0, we have no false alarms for a relatively large sampling interval (600 ms). This is because

1. the low robustness is due to the initial value  $x_1 = 0$ ;
2. initially,  $\dot{x}_1 > 0$  holds in the bounded model, and we have no false alarm there;
3. we potentially have false alarm only around the switching; and
4. around the switching, the robustness is 0.099, which is not very small.

## C Detailed experiment results

Fig. 13 shows the plots of the experiment results omitted from the paper. We note that these results are available in Tables 2 to 4.

## D Detail of the benchmarks

Fig. 14 shows the bounded model of dimension 2 in `ACCD`. The affine hybrid automaton in Fig. 15 represents the original model of `NAV`. The affine hybrid automaton in Fig. 16 represents the original model of `GASBURNER`. Fig. 17 shows the bounded model of dimension 5 in `ACCC`.

## E Omitted Example

**Example 21.** Let  $w$  and  $\mathcal{M}$  be the ones in Example 16. In line 1 of Procedure 1, we let  $State_0 = \{(\ell_0, (40, 35))\}$ . In line 3, we compute the time-bounded reachability analysis and the result is as follows.

$$\begin{aligned}
 State'_1 = & \{(\ell_0, v_1) \mid v_1(x_1) \in [115, 125], v_1(x_2) \in [115, 125]\} \\
 & \cup \{(\ell_0, v_1) \mid -3v_1(x_1) + 11v_1(x_2) \geq 876, -2v_1(x_1) + 9v_1(x_2) \geq 789, \\
 & \quad v_1(x_2) \leq 431/3, v_1(x_1) \leq 499/3, v_1(x_2) \geq 115, \\
 & \quad v_1(x_1) \geq 115, 4v_1(x_1) - 7v_1(x_2) \geq -415\} \\
 & \cup \{(\ell_1, v_1) \mid -3v_1(x_1) + 11v_1(x_2) \geq 876, -v_1(x_1) + 5v_1(x_2) \geq 456, \\
 & \quad -3v_1(x_2) \geq -431, -3v_1(x_1) \geq -499, v_1(x_2) \geq 115, \\
 & \quad v_1(x_1) \geq 115, 4v_1(x_1) - 7v_1(x_2) \geq -415\} \\
 & \cup \{(\ell_2, v_1) \mid -18v_1(x_1) + 15v_1(x_2) \geq -415, -v_1(x_1) + 2v_1(x_2) \geq 115, \\
 & \quad 4v_1(x_1) - 7v_1(x_2) \geq -415, v_1(x_1) \geq 115\} \\
 & \cup \{(\ell_3, v_1) \mid v_1(x_1) \in [115, 455/3], -3v_1(x_1) + 11v_1(x_2) \geq 920, \\
 & \quad v_1(x_2) \leq 415/3, 4v_1(x_1) - 7v_1(x_2) \geq -415\}
 \end{aligned}$$

In line 4, we require  $x_1 = 123$  and  $x_2 = 117$ , and we have  $State_1 = \{(\ell_0, (123, 127)), (\ell_1, (123, 127))\}$ . Since  $\ell_0 \notin F$  and  $\ell_1 \notin F$ , we have  $Result_1 = \emptyset$ .

After incrementing  $i$  in line 2, in line 3, we again compute the time-bounded reachability analysis and the result is as follows.

$$\begin{aligned}
 State'_2 = & \{(\ell_0, v_2) \mid v_2(x_1) \in [198, 253], v_2(x_2) \in [197, 227], \\
 & \quad -2v_2(x_1) + 9v_2(x_2) \geq 1357, 4v_2(x_1) - 7v_2(x_2) \geq -657\} \\
 & \cup \{(\ell_1, v_2) \mid v_2(x_1) \in [233, 253], v_2(x_2) \in [207, 227]\} \\
 & \cup \{(\ell_1, v_2) \mid -3v_2(x_1) + 11v_2(x_2) \geq 1540, -v_2(x_1) + 5v_2(x_2) \geq 784, \\
 & \quad -3v_2(x_2) \geq -673, -3v_2(x_1) \geq -737, v_2(x_2) \geq 197, \\
 & \quad v_2(x_1) \geq 198, 4v_2(x_1) - 7v_2(x_2) \geq -657\} \\
 & \cup \{(\ell_2, v_2) \mid -6v_2(x_1) + 5v_2(x_2) \geq -219, -v_2(x_1) + 2v_2(x_2) \geq 198, \\
 & \quad 4v_2(x_1) - 7v_2(x_2) \geq -657, v_2(x_1) \geq 198\} \\
 & \cup \{(\ell_3, v_2) \mid v_2(x_1) \in [198, 231], v_2(x_2) \leq 219, \\
 & \quad -3v_2(x_1) + 11v_2(x_2) \geq 1584, 4v_2(x_1) - 7v_2(x_2) \geq -657\}
 \end{aligned}$$

In line 4, we require  $x_1 = 203$  and  $x_2 = 201$ . This time, we have  $State_1 = \{(\ell, (203, 201)) \mid \ell \in L\}$ . Thus, we have  $Result_2 = \{(203, 201)\}$ .

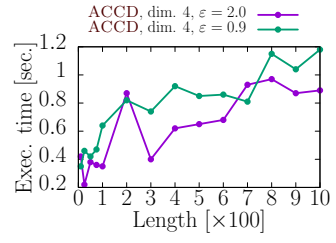
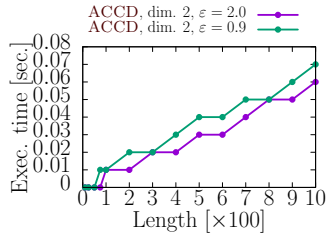


Figure 13: The execution time of HAMONI for ACCD dimension 2 (left) and 4 (right)

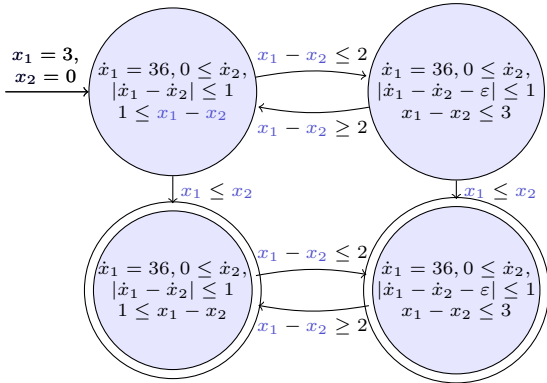


Figure 14: The LHA of dimension 2 in ACCD

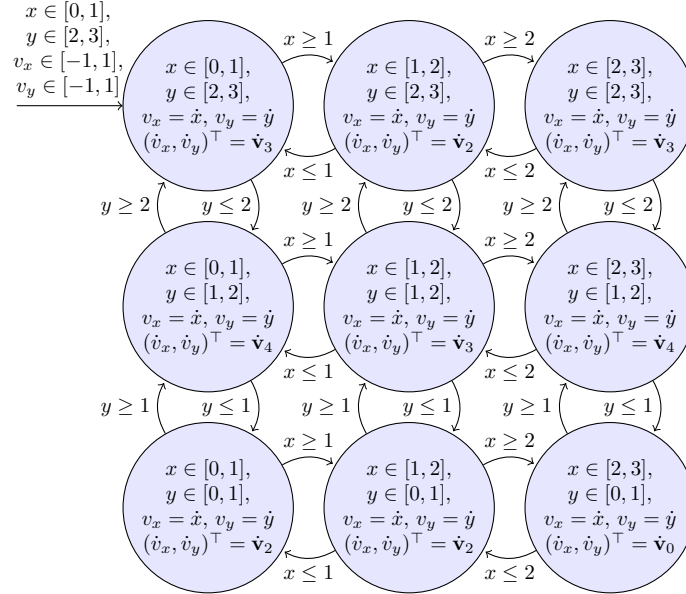


Figure 15: The affine hybrid automaton for the original model in **NAV**, where  $A = \begin{pmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{pmatrix}$ ,  $v_{d,i} = (\sin(i\pi/4), \cos(i\pi/4))^\top$ ,  $\dot{\mathbf{v}}_i = A((v_x, v_y)^\top - v_{d,i})$  for  $i \in \{2, 3, 4\}$ , and  $\dot{\mathbf{v}}_0 = A(v_x, v_y)^\top$

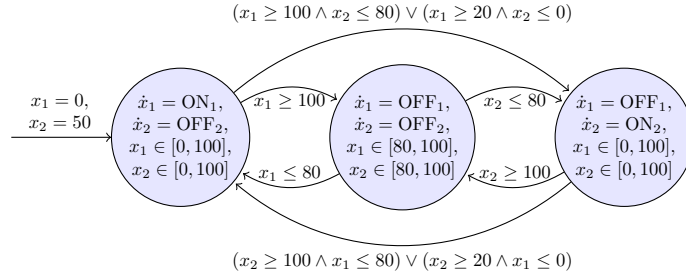


Figure 16: The affine hybrid automaton for the original model in **GASBURNER**, where  $h = 2$ ,  $a = 0.01$ ,  $b = 0.005$ ,  $\text{ON}_1 = h - ax_1 + bx_2$ ,  $\text{ON}_2 = h - ax_2 + bx_1$ ,  $\text{OFF}_1 = -ax_1 + bx_2$ , and  $\text{OFF}_2 = -ax_2 + bx_1$ .

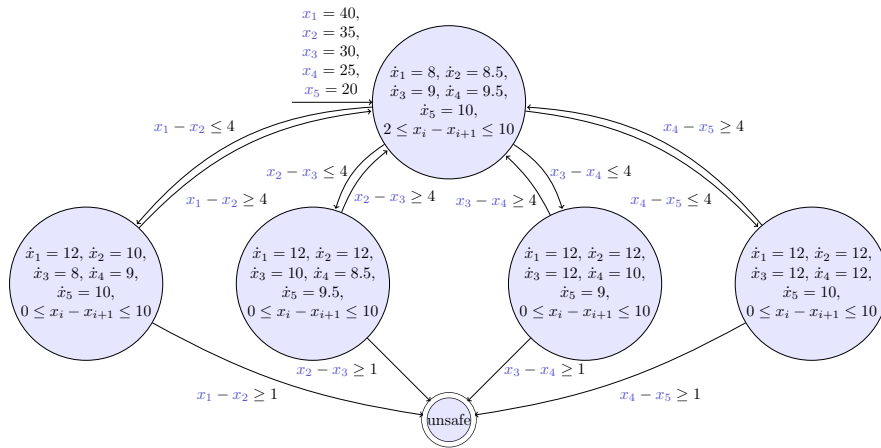


Figure 17: The LHA of dimension 5 in ACCC, where  $i \in \{1, 2, 3, 4\}$