# Sampling Performance of Periodic Event-Triggered Control Systems a Data-driven Approach

Peruffo, Andrea; Mazo, Manuel

**DOI**
10.1109/TCNS.2024.3425646

**Publication date**
2025

**Document Version**
Final published version

**Published in**
IEEE Transactions on Control of Network Systems

# Sampling Performance of Periodic Event-Triggered Control Systems: A Data-Driven Approach

Andrea Peruffo, *Member, IEEE*, and Manuel Mazo Jr., *Senior Member, IEEE*

***Abstract*—In this article, we employ the scenario optimization theory to compute a traffic abstraction, with probability guarantees of correctness, of a periodic event-triggered control (PETC) system with unknown dynamics from a finite number of samples. To this end, we extend the scenario optimization approach to multiclass support vector machines in order to compute a map between the concrete state space and the intersample times of the PETC. This map allows the construction of a traffic abstraction, through an $\ell$-complete relation, that provides upper and lower bounds on the sampling performance of the concrete system. We further propose an alternative path to build such abstraction; first, we identify the model and then apply a model-based procedure. Numerical benchmarks show the practical applicability of our methods for noiseless and noisy samples.**

***Index Terms*—Data-driven modeling, discrete-event systems, system verification.**

## I. Introduction

OVER the past few years, control devices have been immersed into communication networks, begetting the field of networked control systems (NCSs): sensors, controllers, and actuators communicate through a digital network. The control engineer is required to manage both the control actions and the communication needed by (possibly many) controllers without compromising the system's performance. The typical implementation of an NCS considers a periodic sampling of the (analog) plant, whose measurements are passed to the controller, which evaluates the best action in order to optimize a performance cost. Alternatives to this paradigm have been proposed in the form of aperiodic sampling methods, e.g., event-triggered control (ETC) [1], [2] and self-triggered control [3]. These approaches aim at decreasing the communication load while maintaining the desired control performance. One variant is represented by periodic event-triggered control (PETC), where the triggering condition is checked at fixed time intervals, while the measurement is transmitted solely when the condition is verified. This scheme is intuitively convenient; nevertheless, *formally* evaluating the communication benefits has been proposed only recently. The use of formal abstractions for PETC models has been preparatory in this sense. The notion of intersample time (IST)—the time between two consecutive measurements—is the other key ingredient to quantify the sampling performance of an ETC system. We can, in fact, construct several metrics as a function of the ISTs of an ETC system. The average intersample time (AIST) embodies the average sampling rate of the system, thus providing a measure of the expected average communication requirements. Notice that a low AIST means a small time interval between consecutive samples, i.e., it implies a fast sampling and high resource consumption. Formal abstractions that focus on the IST, or rather on *sequences of* ISTs, are called traffic abstractions. In particular, the idea of model-based traffic abstraction has been exploited in, e.g., [4] for linear models and in, e.g., [5] for nonlinear and stochastic models; the tool ETCetera is presented in [6]. The construction of a model-based traffic abstraction entails several challenges. For linear systems, it involves solving a quadratic program over the system's state $x \in \mathbb{R}^n$, which is computationally heavy, especially in high-dimensional systems. For nonlinear systems, no general method exists. The construction of an abstraction usually requires accurate knowledge of the underlying system. However, this knowledge may be unreliable or expensive to attain. We thus resort to probably approximately correct (PAC) methods: this entails obtaining, with high probability, a hypothesis that is a good approximation of an unknown target concept or phenomenon [7]. Data-driven abstractions for dynamical systems have recently gained interest: in [8] and [9], an interval Markov decision process (iMDP) is constructed from samples, and in [10], a behavioral relationship between the abstraction and the system is defined using one-step transitions. All these works provide probabilistic bounds based on *one-step* transitions, whereas our approach embeds the transitions into $\ell$-long sequences, related to abstract states. Furthermore, Coppola et al. [11] discussed PAC bounds for general deterministic systems. In this work, we specialize these findings to PETC systems.

We build upon this literature, aiming at constructing finite-state abstraction of a PETC system with unknown dynamics based on a finite number of samples. By considering sequences of $\ell$ ISTs, we construct $\ell$-complete abstractions, drawing inspiration from the model-based procedure [4]. Our approach

Manuscript received 1 June 2023; revised 27 November 2023 and 8 February 2024; accepted 6 May 2024. Date of publication 9 July 2024; date of current version 20 March 2025. This work was supported by the European Research Council through the SENTIENT Project under Grant ERC-2017-STG #755953. Recommended by Associate Editor K. You. *(Corresponding author: Andrea Peruffo.)*

The authors are with the Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: a.peruffo@tudelft.nl).

Digital Object Identifier 10.1109/TCNS.2024.3425646

is based on the partitioning of the state space according to the $\ell$-sequences, as we exploit the scenario theory to obtain a relation, with PAC guarantees, between the concrete state space and the ISTs. Every abstract state of the traffic model represents an $\ell$-sequence, which is related to a unique region of the concrete space; the transitions among abstract states are governed by the so-called domino rule. The traffic abstraction can be used to compute upper and lower bounds on the AIST; we may refine this procedure by increasing $\ell$, which acts as a tradeoff between precision and computational burden: a higher $\ell$ increases the number of abstract states and transitions and increases the precision of the abstraction. We leverage the scenario theory, which is a computational tool that provides probability guarantees by solving a convex optimization program [12], [13], [14], [15] with a given *confidence level*, which is user defined. In addition, we propose a PAC identification procedure to estimate linear systems' dynamics to ultimately employ the model-based approach [4]. The resulting $\ell$-complete abstraction is then compared against the data-driven one.

This work outlines an approach to provide data-driven models of PETC systems. In particular, we discuss: 1) the data-driven abstraction with PAC guarantees and 2) a two-step procedure composed of model identification (with PAC bounds) followed by a model-based abstraction. These abstractions can be used for several purposes, e.g., to provide upper and lower bounds on the IST, given an initial state, and on the AIST. Naturally, we can employ an abstraction to verify more general properties; for instance, one may analyze when and why a particularly low IST occurs, in order to improve the resource consumption of the concrete system. In this sense, a traffic abstraction is the first step to study ETC systems.

An alternative approach to analyze aperiodic sampling systems is offered in [16] and [17], where the authors compute the maximum sampling interval of an unknown dynamical system and propose techniques to design a controller that favors a large sampling interval via linear-matrix-inequality-based optimization techniques. However, this approach lacks the formal or probabilistic guarantees offered by abstraction-based methods.

*Contributions:* This work is an extended work of [18], where we first introduce data-driven abstractions of deterministic linear PETC systems. In particular, we additionally: 1) discuss a model identification technique with PAC guarantees and devise a consequent model-based abstraction, which is then compared against the data-driven abstraction and the model-based software tool ETCetera [6]; 2) further consider nonlinear models and noisy data; and 3) prove that two formulations of multiclass support vector machine (SVM) are a valid instance of a scenario program.

## II. Preliminaries

### A. PETC Systems

Consider a time-invariant plant controlled with sample-and-hold state feedback described by

$$\dot{\xi}(t) = f(\xi(t), \nu(t)), \quad \nu(t) = K(\hat{\xi}(t)) \tag{1}$$

where $\xi(t) \in \mathbb{R}^{n_x}$ represents the state of the plant whose initial value is $\xi(0) = x_0$, $\hat{\xi}(t) \in \mathbb{R}^{n_x}$ denotes the state measurement that is received by the controller, $\nu(t) \in \mathbb{R}^{n_u}$ indicates the control input, and $n_x$ and $n_u$ are the state-space and input-space dimensions, respectively. The measurements are transmitted only at specific sampling times, with their values being zero-order held on the controller: let $t_i \in \mathbb{R}_+$, $i \in \mathbb{N}_0$, be a sequence of sampling times, with $t_0 = 0$ and $t_{i+1} - t_i > \varepsilon$ for some $\varepsilon > 0$; then, $\hat{\xi}(t) = \xi(t_i)\ \forall\, t \in [t_i, t_{i+1})$. In PETC, a triggering condition is checked every $h$ time instants.

The triggering condition is usually quadratic [19], and we additionally set a maximum intersampling time $\overline{\kappa}$, known as the *heartbeat* of the system. Formally, we define the $(i+1)$th triggering time as

$$t_{i+1} = \inf \left\{ kh > t_i \;\middle|\; \begin{bmatrix} \xi(kh) \\ \xi(t_i) \end{bmatrix}^T Q \begin{bmatrix} \xi(kh) \\ \xi(t_i) \end{bmatrix} > 0 \text{ or } kh - t_i \geq \overline{\kappa} \right\} \tag{2}$$

where $k \in \mathbb{N}$ and $Q \in \mathbb{S}^{2n_x}$ is the designed triggering matrix. A trigger time is thus defined as the first time instant, which is a multiple of $h$ (defined $kh$, with $k > 0$), when the quadratic condition (2) is positive, typically checking a condition as $|\xi(t) - \xi(t_i)|^2 > \sigma^2 |\xi(t)|^2$, for a user-defined parameter $\sigma$. In practical terms, the condition triggers when the difference between the current state $\xi(t)$ and the previous sampled state $\xi(t_i)$ is greater than a given threshold (relative to the state's magnitude). Alternatively, the trigger time is defined as the minimum $kh$ such that the difference $kh - t_i$ is not smaller than $\bar{\kappa}$, meaning that two triggering times cannot be apart for more than $\bar{\kappa}$ time instants. Every run of system (1), starting from the initial condition $\xi(0)$, generates an infinite sequence of samples $\{\xi(t_i)\}$ and of ISTs $\{\tau(\xi(t_i))\}$—denoted $\{\tau_i\}$ for brevity. Implicitly, every IST $\tau_i$ is associated with a sample $\tau(\xi(t_i))$, and it represents the difference between the current and the next sample time, i.e., $\tau(\xi(t_i)) = t_{i+1} - t_i$. In the following, we solely consider PETC settings; hence, $\tau$ is a multiple of $h$, i.e., $\tau \in \{h, 2\,h, \ldots \overline{\kappa} h\}$; for simplicity and without loss of generality, we consider $h = 1$. We name *traffic model* a system described by (1) and (2) whose output is the IST.

Let us define the AIST for every initial state $\xi(0)$ as

$$\text{AIST}(\xi(0)) = \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau(\xi(t_i)). \tag{3}$$

Notice that the regular limit may not exist in pathological cases [20]: for this reason, we employ the liminf instead of lim, thus offering a well-defined metric. Let us characterize the SAIST and LAIST, i.e., *smallest* and the *largest* AIST, respectively, as

$$\begin{aligned} \text{SAIST} &:= \inf_{x_0 \in \mathbb{R}^{n_x}} \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau(\xi(t_i)) \\ \text{LAIST} &:= \sup_{x_0 \in \mathbb{R}^{n_x}} \limsup_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau(\xi(t_i)). \end{aligned} \tag{4}$$

Notice that both the SAIST and LAIST do not depend on an initial state as they are the infimum and supremum over all the possible states $x_0$. The SAIST and the LAIST can be used to assess the performance of an ETC implementation: for instance, in terms of band utilization or general resource consumption. Notice that the computation of (3) and (4) requires to collect infinite ($n \to \infty$) trajectories and the search for a state $x_0$ over

$\mathbb{R}^{n_x}$: both the tasks are computationally hard[1] (if at all possible). To overcome this impediment, we leverage the (finite-state) abstraction theory.

### B. Abstractions and Transition Systems

The construction of a system's abstraction is a computational tool that allows the analysis of large (even infinite) models and, as in our framework, facilitates the evaluation of metrics of interest. In this work, we assume that the *concrete* model is in the form (1) and (2), and we employ a finite-state abstraction in the form of a weighted transition system (WTS).

***Definition 1 (WTS [21], [22]):*** A transition system $\mathcal{S}$ is a tuple $(\mathcal{X}, \mathcal{X}_0, \mathcal{E}, \mathcal{Y}, \mathcal{H}, \gamma)$, where we have the following.

1) $\mathcal{X}$ is the set of states.
2) $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states.
3) $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of edges or transitions.
4) $\mathcal{Y}$ is the set of outputs.
5) $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is the output map.
6) $\gamma : \mathcal{E} \to \mathbb{Q}$ is the weight function.

We remark that the definition in [21] includes an action set $\mathcal{U}$: this is neglected since we consider autonomous systems; furthermore, the cardinality of $\mathcal{X}$ is finite as we consider finite-state models. We tacitly assume *nonblocking* transition systems, i.e., automata where every state is equipped with at least one outgoing transition. The WTS is endowed with nondeterminism, i.e., states may have multiple outgoing transitions, but no probability measure is laid upon them. Model (1) and (2) can be embedded as a transition system, leaving its state set, initial state set, output set, and output map unchanged, where $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of all pairs $x, x'$ s.t. $\mathcal{E} = \{(x, x') \in \mathcal{X} \times \mathcal{X} : x' = f(x)\}$. The embedding of a system does not affect the set of infinite (or finite) behaviors.

Let us define $r = \mathtt{x}_0\mathtt{x}_1\mathtt{x}_2 \ldots$, an infinite internal behavior, or *run*, of $\mathcal{S}$ if $\mathtt{x}_0 \in \mathcal{X}_0$ and $(\mathtt{x}, \mathtt{x}') \in \mathcal{E}$ for all $i \in \mathbb{N}$, and $\mathcal{B}(r) = \mathtt{y}_0\mathtt{y}_1\mathtt{y}_2 \ldots$, its corresponding *external* behavior, or trace, if $\mathcal{H}(\mathtt{x}_i) = \mathtt{y}_i$ for all $i \in \mathbb{N}$. The set of all infinite external behaviors of $\mathcal{S}$ starting from state $\mathtt{x}$ is denoted by $\mathcal{B}^\omega(\mathtt{x})$. Similarly, we define $\gamma(r) = v_0 v_1 \ldots$ as the sequence of weights from run $r$, where $\gamma(\mathtt{x}, \mathtt{x}') = v$. For simplicity, we consider the state output map being equal to the weight function, i.e., $\gamma(\mathtt{x}, \mathtt{x}') = \mathcal{H}(\mathtt{x})$ for all $\mathtt{x}'$. Recall that we are interested in evaluating the AIST of the concrete system: to this end, we define a value function, i.e., a function mapping an infinite sequence of weights to a (possibly finite) value, as

$$\mathrm{LimAvg}(\gamma(r)) := \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} v_i. \tag{5}$$

A weighted automaton equipped with the LimAvg value function is called a LimAvg automaton [22]. Similarly, we can consider

$$\begin{aligned} \overline{V}(\mathcal{S}) &:= \sup\{\mathrm{LimAvg}(\gamma(r)) \mid r \text{ is a run of } \mathcal{S}\} \\ \underline{V}(\mathcal{S}) &:= \inf\{\mathrm{LimAvg}(\gamma(r)) \mid r \text{ is a run of } \mathcal{S}\}. \end{aligned} \tag{6}$$

It is easy to notice that $\underline{V}(\mathcal{S})$ and $\overline{V}(\mathcal{S})$ are, respectively, lower and upper bounds of LimAvg $\forall r$. Remarkably, the authors of [22] and [23] show that we can evaluate $\underline{V}(\mathcal{S})$ and $\overline{V}(\mathcal{S})$ of a WTS in polynomial time via a cycle analysis of the underlying graph. Infinite behaviors of a transition system are related to *cycles:* in fact, the first value function in (6) evaluates to the *smallest average cycle* (SAC) of the transition system. In many contexts, the performance of systems as the (minimum) average resource consumption is modeled as the minimum cycle mean problem. Conversely, we may be interested in the *largest* average cycle (LAC) that outlines the worst case resource consumption. The smallest and largest cycles given an initial state $\mathtt{x}_0$ are denoted as $\underline{V}(\mathcal{S}(\mathtt{x}_0))$ and $\overline{V}(\mathcal{S}(\mathtt{x}_0))$, respectively.

[1]A large $n$ can be used to approximate (3), however, without guarantees on this value representing an over- or underapproximation of the actual AIST.

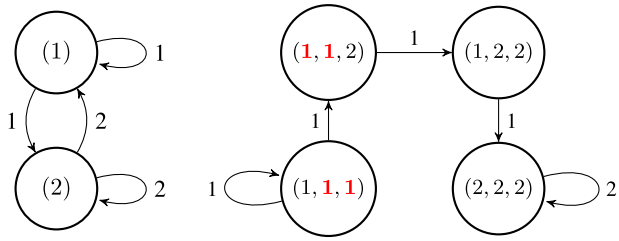



Fig. 1. S$\ell$-CA with $\tau_i = \{1, 2\}$ and $\ell = 1$ (left), $\ell = 3$ (right). The states represent $\ell$-sequences of ISTs. The domino rule is highlighted for transition $(1, \mathbf{1}, \mathbf{1}) \to (\mathbf{1}, \mathbf{1}, 2)$.

Equations (5) and (6) clearly mimic (3) and (4), as we prepare to design a transition system that abstracts a PETC model. We introduce the notion of *behavioral inclusion:* the abstraction does not require any knowledge of the internal behavior of a system, but the external traces exhibited by the concrete system must be included in the abstraction's possible behaviors.

***Definition 2 (Behavioral inclusion [21]):*** Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\mathcal{Y}_a = \mathcal{Y}_b$. We say that $\mathcal{S}_b$ behaviorally includes $\mathcal{S}_a$, denoted by $\mathcal{S}_a \preceq_{\mathcal{B}} \mathcal{S}_b$, if $\mathcal{B}^\omega(\mathcal{S}_a) \subseteq \mathcal{B}^\omega(\mathcal{S}_b)$. □

In practical terms, every possible trace of the traffic model must be included in a run of the WTS.

In order to build an abstraction of a traffic model, the intuition suggests to relate each output symbol $\tau_i$ to an abstract state of the abstraction $\mathtt{x}_i$. With this in mind, we employ the so-called $\ell$-complete model.

***Definition 3 (Strongest $\ell$-complete abstraction [4]):*** Let $\mathcal{S} := (\mathcal{X}, \mathcal{X}_0, \mathcal{E}, \mathcal{Y}, \mathcal{H})$ be a transition system, and let $\mathcal{X}_\ell \subseteq \mathcal{Y}^\ell$ be the set of all $\ell$-long subsequences of all behaviors in $\mathcal{S}$. Then, the system $\mathcal{S}_\ell = (\mathcal{X}_\ell, B_\ell(\mathcal{S}), \mathcal{E}_\ell, \mathcal{Y}, \mathcal{H})$ is called the strongest $\ell$-complete abstraction (S$\ell$-CA) of $\mathcal{S}$, where we have the following.

1) $\mathcal{E}_\ell = \{(k\sigma, \sigma k') \mid k, k' \in \mathcal{Y}, \sigma \in \mathcal{Y}^{\ell-1}, k\sigma, \sigma k' \in \mathcal{X}_\ell\}$.
2) $\mathcal{H}(k\sigma) = k$.

Here, we denote by $B_\ell(\mathcal{S})$ all the possible external traces of system $\mathcal{S}_\ell$, and $\mathcal{Y}^\ell$ is the Cartesian product $\mathcal{Y} \times \cdots \times \mathcal{Y}$ repeated $\ell$ times. The S$\ell$-CA relates each abstract state as an $\ell$-long external trace. Its transitions follow the so-called "domino rule": e.g., let us assume $\ell = 3$ and let us observe a trace *xyz* (recall that every $\ell$-trace corresponds to a single abstract state); the next $\ell$-trace must begin with *yz*. Therefore, a transition from *xyz* can lead to, e.g., *yzx*, *yzy*, and *yzz*. The output of an abstract state is simply the first component of the $\ell$-sequence: state *xyz* returns $\mathcal{H}(xyz) = x$.

The S$\ell$-CA translation solely employs the external behavior, i.e., the sequences of $\tau_i$, to build the abstraction. Two examples of S$\ell$-CA, where $\tau_i = \{1, 2\}$, are shown in Fig. 1. Not all the

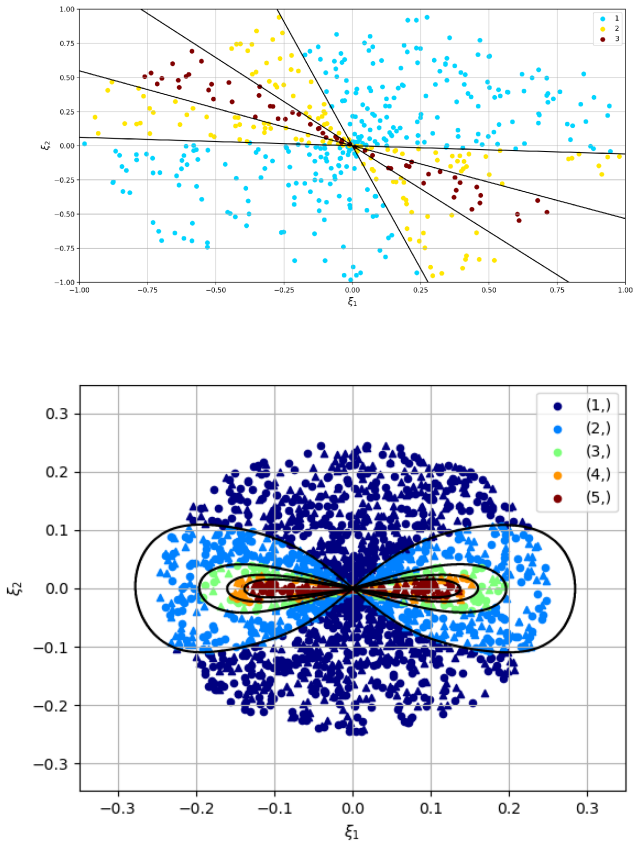



Fig. 2. IST regions for a 2-D linear (top, see [18]), nonlinear (bottom) models. Black lines denote the regions' boundaries, depicted along with 1000 samples in different colors, for $\ell = 1$.

possible three sequences of ISTs are present in the S$\ell$-CA: this is often the case, as the triggering condition [see (2)] defines a particular time behavior of the PETC system. In other words, solely a *subset* of all possible $\ell$-sequences is allowed by the triggering condition. Model-based works [4] present an abstraction of (linear) systems in the form (1) and (2) as an $\ell$-complete abstraction. Although dynamical systems (1) and (2) have an infinite state space, they have a finite-cardinality output, i.e., the IST, which belongs to $[1, \bar{\kappa}]$.

### C. Linear Systems and Veronese Embedding

As discussed in [4], each $\ell$-sequence of ISTs corresponds to a union of cones, when considering linear systems (shown in Fig. 2). As such, we can translate the samples into conic coordinates to get a well-suited representation. To this end, we employ the kernel trick [24]: samples are embedded into a higher dimensional space, where it is possible to find hyperplanes to linearly separate the dataset. As the state space of linear systems is partitioned into cones, we choose a conic transformation known as the Veronese map [25] of order 2, which maps into all possible monomials of total order 2, $\mathcal{V}_2 : \mathbb{R}^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$, defined by

$$\mathcal{V}_2(\mathbf{v}) := [v_1^2 \; v_2^2 \; \dots \; v_n^2 \quad v_1 v_2 \; \dots \; v_{n-1} v_n] \tag{7}$$

where $v_i$ indicates the $i$th component of vector $\mathbf{v}$. This embedding is related to a homogeneous polynomial kernel: the latter presents the bilinear terms multiplied by $\sqrt{2}$. The two embeddings thus project vectors into the same feature space. Furthermore, the conic partitions fulfill an *encapsulation* constraint [4]: the cone corresponding to $\tau = j + 1$ is *within* the region corresponding to $\tau \geq j$, for all $j$. Formally, for all $\xi(\cdot) = x$ such that $\mathcal{T}(x) = j$, the following holds:

$$x^T N_i x < 0, \quad \text{for} \quad i < j, \quad x^T N_j x > 0 \tag{8}$$

where the set of matrices $N_i$ is a function of $A$, $B$, $K$, and $Q$ [see (1) and (2)]. The actual expression of $N_i$ is omitted for brevity, and it can be found in [4]. Intuitively, $N_i$ indicates a violation of the $i$th triggering condition. For instance, let us assume a point $x$ satisfying

$$x^T N_1 \, x < 0, \quad x^T N_2 \, x > 0. \tag{9}$$

The triggering condition is not violated after one IST, but it is violated after two ISTs. Hence, the next IST corresponding to $x$ is 2, i.e., $\mathcal{T}(x) = 2$.

We rewrite inequalities (8) by applying the Veronese map to all samples $x$ and obtain a set of linear constraints

$$W_i \cdot \mathcal{V}_2(x) < 0, \quad \text{for } i < j, \quad W_j \cdot \mathcal{V}_2(x) > 0 \tag{10}$$

where $W_i$—that are related to $N_i$ through an algebraic transformation—shall be recovered from the samples $Z_i$. Constraints (10) can trivially be rewritten as a multiclass SVM approach. Nonlinear systems, in general, might show rather complex partitions' shape; in this case, we employ a radial basis function (RBF) kernel, commonly used in several SVM applications.

### D. Multiclass SVM

Several formulations of multiclass SVM exist in the literature: to ease the presentation, we choose a Crammer–Singer [26] SVM formulation; other formulations are discussed in Appendix C. This formulation consists of one convex problem considering $L$ (one per class) hyperplanes. Let us define the function

$$h_j(z) := W_j \, z + b_j, \quad \text{for} \quad j \in \{1, \dots, L\} \tag{11}$$

where $W_j \in \mathbb{R}^{n \times 1}$, $b_j \in \mathbb{R} \; \forall \, j \in \{1, \dots, L\}$. The classification decision follows the *winner-takes-all* policy; formally

$$D(z) := \arg\max_j \{h_j(z)\}. \tag{12}$$

In other words, the SVM algorithm assigns label $D(\mathbf{X}_i)$ to each sample $\mathbf{X}_i$. A correct classification occurs whenever the $\mathbf{Y}_i$th function, denoted with parameters $(W_{\mathbf{Y}_i}, b_{\mathbf{Y}_i})$, takes the largest value, i.e., $D(\mathbf{X}_i) = \mathbf{Y}_i$. Equivalently, we impose

$$\zeta - (W_{\mathbf{Y}_i} - W_j)\mathbf{X}_i - (b_{\mathbf{Y}_i} - b_j) \leq 0 \quad \forall j \neq \mathbf{Y}_i \tag{13}$$

where the hyperparameter $\zeta > 0$ is added for numerical stability. Let us define

$$g(\mathbf{X}_i, \mathbf{Y}_i) := \max_{j \neq \mathbf{Y}_i} \{\zeta - (W_{\mathbf{Y}_i} - W_j)\mathbf{X}_i - (b_{\mathbf{Y}_i} - b_j)\}. \tag{14}$$

Hence, conditions (13) can be summarized as

$$g(\mathbf{X}_i, \mathbf{Y}_i) \leq 0. \tag{15}$$

Constraints $g(\mathbf{X}_i, \mathbf{Y}_i) \leq 0$ can be satisfied only when the dataset is linearly separable. To further accommodate for more

general cases—e.g., a noisy dataset—we establish a *constraint violation*, i.e., we allow $g(\mathbf{X}_i, \mathbf{Y}_i) > 0$ for a possibly limited number of samples. To this end, we employ slack variables $\theta_i \geq 0$, $i = 1, \ldots, N$, and rewrite constraints (15) as $g(\mathbf{X}_i) \leq \theta_i$. An $L$-class classifier can be obtained by solving the following convex program:

$$\min_{W,b,\theta_i \geq 0} \quad \sum_{j=1}^{L} ||W_j||^2 + \rho \sum_{i=1}^{N} \theta_i,$$
$$\text{s.t.} \quad g(\mathbf{X}_i, \mathbf{Y}_i) \leq \theta_i, \quad i = 1, \ldots, N \tag{16}$$

where $\rho > 0$ is a hyperparameter balancing the tradeoff between the correctness of the algorithm (the number of positive $\theta_i$) and its cost (the norm of matrices $W_j$). Note that the minimization of $\|W_j\|$ is related to maximizing the distance between samples and the hyperplane in order to improve the classification performance.

### E. Nonlinear Dynamics

This approach can be extended to nonlinear systems: each $\ell$-sequence corresponds to a partition (see Fig. 2) whose shape depends on the nonlinearity of the underlying system: we then use the RBF kernel, capable to generalize to arbitrary shapes, in place of the Veronese embedding. In practice, this is implemented solely for binary classifiers, and multiclass problems are reduced to a multitude of binary programs. We outline our approach considering a one-vs-all formulation drawing inspiration from the chain-of-classifiers [27] approach for multilabel problems, as the encapsulation constraints align with this setting. We set $L$ binary classifiers; for each classifier $\mathcal{C}_j$, for $j = 1, \ldots L$, we relabel $\mathbf{Y}_i$ as

$$\mathbf{Y}'_i = \begin{cases} -1, & \text{if } \mathbf{Y}_i \leq j \\ +1, & \text{otherwise.} \end{cases} \tag{17}$$

In practical terms, $\mathcal{C}_1$ separates the samples corresponding to $\mathbf{Y}_i = 1$ from $\mathbf{Y}_i \geq 2$, $\mathcal{C}_2$ separates $\mathbf{Y}_i = \{1, 2\}$ from $\mathbf{Y}_i \geq 3$, and so on. The chain of classifiers resembles the encapsulation constraints, as we know that the partition corresponding to $\mathbf{Y}_i = 3$ is within the partition $\mathbf{Y}_i = 2$, which is, in turn, within the partition $\mathbf{Y}_i = 1$.

## III. Data-Driven Abstractions With Guarantees

### A. Scenario Guarantees for Multiclass SVM

Once a classifier is found, we are interested in quantifying its reliability, i.e., to estimate the chance that the labels it returns are correct or incorrect. To this end, we employ the concept of risk (or violation probability), which evaluates the probability that a (new, unseen) sample violates its constraint.

***Definition 4 (see [15]):*** The probability (risk) of violating a constraint $g(\mathbf{X}, \mathbf{Y})$, given any randomly generated datum $(\mathbf{X}, \mathbf{Y})$, is denoted as

$$R(\mathfrak{S}) := \mathbb{P}[g(\mathbf{X}, \mathbf{Y}) > 0] \tag{18}$$

where $g(\mathbf{X}, \mathbf{Y})$ is defined according to (15) and where we denote as $\mathfrak{S}$ the parameters of the SVM algorithm $(W_j, b_j)$ for all $j = 1, \ldots L$.

Let us outline three assumptions required to define a proper scenario program.

***Assumption 1 (Mathematical setup):*** Let us denote $\mathfrak{S}$, an element of a vector space $\mathcal{W}$, as the hyperplane parameters, i.e., $\mathfrak{S} = \{(W_i, b_i) \text{ for } i = 1, \ldots L\}$. The cost function $c(\mathfrak{S})$ and, for any given $\mathbf{X} \in \mathcal{X}$, $g(\mathfrak{S}, \mathbf{X})$ are convex functions of $\mathfrak{S}$. The scenarios $\mathbf{X}_i$, $i = 1, \ldots, N$, form an independent random sample from $(\mathcal{X}, \mathcal{G}, \mathbb{P})$.

***Assumption 2 (Existence and uniqueness):*** Consider optimization problems as in (16), where $N$ is substituted with any index $m = 0, 1, \ldots$, and $\mathbf{X}_i, i = 1, \ldots, m$, is an independent sample from $(\mathcal{X}, \mathcal{G}, \mathbb{P})$. For every $m$ and for every outcome of $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m)$, we assume that each optimization problem admits a solution (i.e., the problems are feasible and the infimum is achieved on the feasibility set). If for one of these optimization problems, more than one solution exists, we shall use a convex tie-break rule, which breaks the tie by minimizing an additional number of convex functionals $t_1(\mathfrak{S}), t_2(\mathfrak{S}), \ldots$.

***Assumption 3 (Nonaccumulation):*** For every hyperplane parameters $\mathfrak{S} = (W, b)$ in $\mathcal{W}$, $\mathbb{P}[g(\mathfrak{S}, \mathbf{X}) = 0 \mid \mathbf{X}] = 0$, where $g(\mathfrak{S}, \mathbf{X})$ is the constraint condition defined in (15).

These three assumptions hold for our optimization program, as outlined in Appendix A. Hence, we can leverage the scenario theory results and provide a quantitative evaluation of the risk of program (16).

***Theorem 1:*** Given a confidence parameter $\beta \in [0, 1]$ and $N$ samples, under Assumptions 1–3, it holds that

$$\mathbb{P}^N[\underline{\epsilon}(s^*, N, \beta) \leq R(\mathfrak{S}^*) \leq \overline{\epsilon}(s^*, N, \beta)] \geq 1 - 3\beta \tag{19}$$

where $\mathfrak{S}^*$ represents the parameters that minimize program (16) and $s^*$ identifies the number of violated constraints, i.e., the number of samples that return $g(\mathbf{X}_i, \mathbf{Y}_i) > 0$. $\mathfrak{S}^*$ depends on the collected $N$ samples and is, thus, a random vector defined over $(\mathcal{X}^N, \mathbb{P}^N)$, where $\mathbb{P}^N$ is the product probability of $N$ independent samples. The bounds $\underline{\epsilon}$ and $\overline{\epsilon}$ are directly proportional to $s^*$ and inversely proportional to $N$ and have a logarithmic dependence on $\beta$. They are computed by solving a polynomial equation; the interested reader may refer to [15] for a detailed discussion. □

***Remark 1:*** Let us underline that the violation of a constraint in (16) does not entail misclassification: $g(\mathbf{X}_i, \mathbf{Y}_i) > 0$ symbolizes that the evaluation of (11) for some $j$ differs less than $\zeta$, which intuitively embodies the minimum permissible distance between two hyperplanes. A given sample might violate its constraint but could be correctly classified: thus, misclassification occurs more rarely than constraint violation. As such, the upper bound on $R(\mathfrak{S}^*)$ provided by Theorem 1 in fact bounds the probability of misclassification.

***Theorem 2:*** Under the same assumptions as Theorem 1, the event of misclassification can be bounded

$$\mathbb{P}^N[\,\mathbb{P}[\text{misclassification}] \leq \overline{\epsilon}(s^*, N, \beta)\,] \geq 1 - 3\beta \tag{20}$$

where the bound $\overline{\epsilon}$ can be found as per Theorem 1. □

***Remark 2 (Nonlinear dynamics):*** The scenario bounds are linked to a *single* program; hence, we compute the violated constraints $s^*$ and the bound $\overline{\epsilon}$ *for every binary classifier*.

### B. Data-Driven Abstractions

Scenario-based methods are based on random samples. When considering a dynamical system, the procedure oftentimes entails sampling (with a given probability distribution) the initial state of a system, lets it run, and collects the trajectory that arises. Similarly, in this work, we randomly sample $x_0 \sim \mathcal{P}$, initialize system (1) and (2), and observe the data point $Z_i = (\mathbf{X}_i, \mathbf{Y}_i)$, for $i \leq N$, where $\mathbf{X}_i = x_0$ and $\mathbf{Y}_i$ is the $\ell$-sequence of ISTs stemming from $x_0$. In practice, this is equivalent to sampling a trajectory of length $\ell$ and recording the corresponding ISTs . As such, data points $Z_i$ hinge on the concrete model *as much as* they depend on the probability distribution used to sample the initial states: these two elements are, in fact, indissoluble from the samples.

We introduce the notion of *probabilistically initialized* (PI) model as follows.

***Definition 5:*** Given a probability distribution $\mathcal{P}$ defined over $\mathcal{X}_0$, a PI model $\mathcal{M}$ is a tuple $(\mathcal{X}, \mathcal{X}_0^{\mathcal{P}}, \mathcal{Y}, \mathcal{H})$, where we have the following.

1) $\mathcal{X}$ is the set of states.
2) $\mathcal{X}_0^{\mathcal{P}}$ is the set of initial states endowed with $\mathcal{P}$.
3) $\mathcal{Y}$ is the set of outputs.
4) $\mathcal{H}$ is the output map.

Here, we consider the input–output relation $\mathcal{X} \to \mathcal{Y}$ to be deterministic. □

A PI model is the actual probabilistic object that originates our dataset. The results stemming from the samples of a PI model bear the name *data-driven*, as the following data-driven S$\ell$-CA.

***Definition 6 (Data-driven S$\ell$-CA [11]):*** Let $\mathcal{M}$ be a PI model and let us collect data $Z$ from it. We define a data-driven S$\ell$-CA $\mathcal{S}_\ell^N = (\hat{\mathcal{X}}_\ell, \mathcal{T}(x_0), \mathcal{E}_\ell, \mathcal{Y}, \mathcal{H})$, where we have the following.

1) $\hat{\mathcal{X}}_\ell$ is the state space built upon $Z$.
2) $\mathcal{T}$ is a classification map with scenario guarantees.

Here, $x_0$ is the concrete system initial state. □

In practical terms, $\hat{\mathcal{X}}_\ell$ is composed of $L$ labels collected within $Z$. As the model-based S$\ell$-CA behaviorally includes the concrete system, the data-driven S$\ell$-CA behaviorally includes the PI model within some probability bounds. Recall that behavioral inclusion implies that *all* traces from the concrete model are within the outputs of the abstraction, i.e., we must ensure that $\mathcal{X}_0^{\mathcal{P}} = \mathcal{X}$.

***Domino completion:*** The structure of the domino transitions coupled with the sample-based states may give rise to a *blocking* transition system. Let us assume a PETC model with ISTs $\tau \in \{1, 2\}$, and let us consider $\ell = 3$. Let us also consider a dataset that exhibits solely the sequences $\{(1,1,1), (1,1,2), (1,2,1)\}$. As the dataset does not include any sequence starting with (2,1), the state corresponding to (1,2,1) has no outgoing transitions, as depicted in Fig. 3. The existence of the sequence (1,2,1) *implies* the existence of at least one sequence starting with (2,1). To overcome this impediment, we artificially add *all* states corresponding to sequences $(2, 1, *)$. We repeat the procedure until we obtain a nonblocking transition system. □

The scenario approach allows us to build a *data-driven abstraction* that stems from the collected samples and the sampling process. We regard the underlying model as a *data-generating process:* in fact, we construct an abstraction of the data-generating process rather than the model per sé. In practical terms, the probability bounds $\bar{\epsilon}$ and the confidence $\beta$ hold for any new sample $x_0$ arising from the same sampling process used to construct the SVM. We use $L$ labels that are the *witnessed* labels from samples $Z_i$: these might be just a portion of all labels. The implications of this are discussed in the following section.

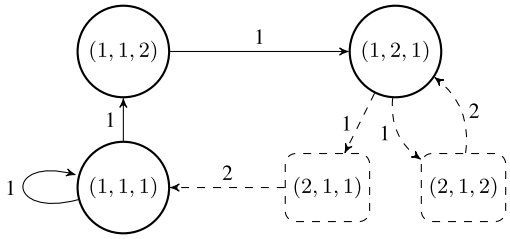



Fig. 3. Construction of a nonblocking automaton. Dashed lines indicate artificial states and transitions.

### C. Unseen Labels and Infinite Behaviors

The use of an abstraction goes beyond the simple collection of finite trajectories: it generates infinite behaviors and we (aim to) use it to verify properties for infinite horizon.

Recent work [11] tackles this issue, which is summarized in the following. Let us assume that the control stabilizes the system's trajectories over some periodic orbit (in terms of ISTs); in other words, we expect the system's behaviors consisting of two components: a transient (of length $H_t$) and a periodic (of length $H_p$) behavior. Furthermore, let us sample trajectories of the concrete system of length $H = H_t + H_p$, and let us split them into $\ell$-(sub)sequences. We construct an S$\ell$-CA to extend the scenario guarantees to infinite horizon as follows. First, let us introduce the notion of probabilistic behavioral inclusion.

***Definition 7 (Probabilistic behavioral inclusion [11]):*** Consider two systems $\mathcal{S}_a$ and $\mathcal{S}_b$ with $\mathcal{Y}_a = \mathcal{Y}_b$. We say that $\mathcal{S}_a$ is behaviorally included in $\mathcal{S}_b$ with probability greater than or equal to $1 - \epsilon$, denoted by $\mathbb{P}[\mathcal{S}_a \preceq_{\mathcal{B}} \mathcal{S}_b] \geq 1 - \epsilon$, if for $x_0 \sim \mathcal{P}$, it holds that

$$\mathbb{P}\left[\mathcal{B}^\omega(\mathcal{S}_a(x_0)) \subseteq \mathcal{B}^\omega(\mathcal{S}_b) \mid x_0\right] \geq 1 - \epsilon \tag{21}$$

where $\mathcal{S}_a(x_0)$ denotes the internal behavior of system $\mathcal{S}_a$ starting from $x_0$.

In practical terms, Definition 7 considers the probability of sampling an initial state $x_0$, from system $\mathcal{S}_a$, such that the behavior stemming from $x_0$ is included in the set of behaviors of $\mathcal{S}_b$.

***Proposition 1 (Adapted From [11]):*** Assume that a PI model $\mathcal{M}$, based on (1) and (2), exhibits periodic behaviors of length $H_p$ after a transient of length $H_t$. Given a set of $N$ trajectories of length $H = H_t + H_p$ collected from $\mathcal{M}$, let us compute an SVM map $\mathcal{T}$ as (16). Consider the data-driven S$\ell$-CA $\mathcal{S}_\ell^N$ constructed from $N$ trajectories of length $H$; then, given a new initial state $x_0$ sampled from $\mathcal{X}_0^{\mathcal{P}}$

$$\mathbb{P}^N[\mathbb{P}[\mathcal{M} \preceq_{\mathcal{B}} \mathcal{S}_\ell^N] \geq 1 - \bar{\epsilon}(s^*, N, \beta)] \geq 1 - 3\beta \tag{22}$$

i.e., with confidence $1 - 3\beta$, $\mathcal{S}_\ell^N$ behaviorally includes the PI model $\mathcal{M}$ with probability not smaller than $1 - \bar{\epsilon}$. □

Proposition 1 provides a time horizon for the sampled trajectories based on the transient and periodic behaviors of the concrete

system. In practical terms, this assumption allows us to observe all the possible (current and future) behaviors of the system in finite time, i.e., within $H$ time steps. Notably, systems abiding the assumption of Proposition 1 admit a bisimulation [11]—this can be proved by considering the bisimulation algorithm in [21] and checking that for periodic patterns, the algorithm terminates in a finite number of steps (for a more detailed discussion, refer to [11]). New evidence suggests that linear PETC systems having periodic behaviors are a rare occurrence, especially as the value of $h$ [see (2)] increases, although no comprehensive relation between system parameters and patterns has been found. Furthermore, Postoyan et al. [20] linked the presence of periodic patterns to a small value of $h$ and to the absence of complex-conjugate eigenvalues of the matrix $A + BK$. The characterization of PETC systems needs to be additionally developed in order to define which systems may exhibit periodic IST patterns.

The bounds in (22) are derived from the SVM formulation. Our procedure builds an abstraction from collected sequences of IST: this implies that the abstraction might lack some states ($\ell$-sequences) that have not been collected within the dataset—the scenario approach provides a bound on the probability of this event. The scenario theory guarantees that the misclassification probability is bounded (see Theorem 2): this holds when the correct label has been previously sampled or not. Hence, the probability of sampling a state with an unseen label is upper bounded by $\bar{\epsilon}$, i.e., the abstraction built on the SVM dataset inherits the same PAC bounds.

### D. Abstraction Precision

Once program (16) is solved, we use the SVM results to assign to any (new) sample $x_0$ a label, which corresponds to an abstract state $\mathrm{x}_0$. We use $\mathrm{x}_0$ as the initial state of the data-driven S$\ell$-CA and thus compute SAC($\mathrm{x}_0$) and LAC($\mathrm{x}_0$)—see (6)—as bounds for AIST($\xi(0)$). The gap between these two values offers additional information as

$$\delta_{\text{AIST}}(\mathrm{x}_0) := \text{LAC}(\mathrm{x}_0) - \text{SAC}(\mathrm{x}_0)$$

defines the precision of our abstraction. Under the probability bounds of Theorem 1, the true AIST dwells within the interval $\delta_{\text{AIST}}$; therefore, the tighter $\delta_{\text{AIST}}$ is, the more precise the information we gather. If $\delta_{\text{AIST}}$ is unsatisfactory, we may increase $\ell$ to refine the abstraction. In this sense, $\ell$ can be seen as a tradeoff: increasing $\ell$ provides a more precise abstraction, with a larger state space and a greater number of labels, which increases the computational time of the SVM algorithm. Furthermore, in order to provide a global precision of the S$\ell$-CA, we define $\hat{\delta}$ as the average $\delta_{\text{AIST}}$ over all states of the abstraction; formally

$$\hat{\delta} := \frac{1}{|\mathcal{X}_\ell|} \sum_{\mathrm{x} \in \mathcal{X}_\ell} \delta_{\text{AIST}}(\mathrm{x}). \tag{23}$$

The $\hat{\delta}$ metric embodies the degree of uncertainty in the evaluation of the AIST for every possible state of the S$\ell$-CA. A smaller $\hat{\delta}$ ensures a more precise abstraction.

The overall procedure, from the classification algorithm up to the bound computation (23), is outlined in Algorithm 1. We can think of the proposed approach as an iterative procedure: for each value of $\ell$, we evaluate (23). If we are satisfied with its value, the procedure stops; if not, we can increase $\ell$ and restart the procedure. To ensure termination, we assume that the user inputs a maximum sequence length $\bar{\ell}$: upon reaching $\ell = \bar{\ell}$, the procedure returns $\hat{\delta}$.

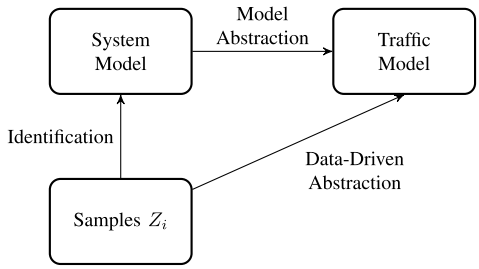



Fig. 4. Two alternative paths from samples to traffic abstraction.

**Algorithm 1:** Data-Driven AIST With PAC Guarantees

**Input:** $\beta$, $\delta$, $(\mathbf{X}, \mathbf{Y})$, $\xi(0)$, $\bar{\ell}$
**Output:** EAC
Collect $N$ samples $Z = \{\mathbf{X}, \mathbf{Y}\}$
$\mathcal{T}, \bar{\epsilon}(s^*, N, \beta) \leftarrow$ SVM($\mathbf{X}$, $\mathbf{Y}$)
Build S$\ell$-CA $\mathcal{S}$
EAC $\leftarrow$ Compute (23)
**if** EAC $< \delta$ or $\ell = \bar{\ell}$ **then**
  **return** EAC
**else**
  $\ell \leftarrow \ell + 1$
**end if**

## IV. Model Identification and Abstractions

Insofar, our procedure builds a traffic abstraction directly from a collection of samples $Z_i = (\mathbf{X}_i, \mathbf{Y}_i)$. Schematically, Fig. 4 depicts two possible approaches to use data to provide quantitative evaluations regarding an unknown system. One way offers the data-driven abstraction, where we attach probability bounds to the correctness of the SVM and consequently to the results regarding the AIST performance. The second way regards a model identification approach, where we use data to identify a model—here, we attach probabilistic bounds to the identified matrices. Then, at the second stage, we use the identified matrices and perform a standard model-based procedure with state-of-the-art tools [6]. We discuss the identification of a linear model, consistently with the literature, with probabilistic guarantees of correctness, once again employing a scenario-based optimization program. This approach can be used in the presence of nonlinear systems, provided that we know the concrete model's class.

### A. PETC Identification With Correctness Guarantees

As noted in [4], the system described by (1) and (2) can be rewritten solely considering the sampled states $\hat{\xi}(t) = \xi(t_i)$. In particular, the event-driven evolution of sampled states can be compactly expressed by

$$\xi(t_{i+1}) = M(\tau(\xi(t_i)) \cdot \xi(t_i) \tag{24}$$

where the matrix $M(\tau)$ is a function of the IST $\tau(\xi(t_i)) = k$, as

$$M(k) := A_d(k) + B_d(k)K = e^{Ak} + \int_0^k e^{Bu} du\, BK. \quad (25)$$

The model-based abstraction proposed in [4] can be computed using the matrices $M(k)$ and $Q$ [see (2)], without accessing the system's dynamics $A$, $B$, and $K$ in (1).

Given a dataset of samples $(x_i, \tau_i, x_{i+1})$, we can solve the linear program associated with (24) to recover the $M(\tau)$ matrices. Deterministic systems allow an exact identification: every matrix $M(\tau)$ can be determined using $n_x$ (linearly independent) samples; thus, $L$ matrices can be identified with $n_x \cdot L$ samples. Notice that we need an augmented sample set $Z_i = (\mathbf{X}_i, \mathbf{Y}_i, \mathbf{X}'_i)$ to describe (24), where $\mathbf{X}'_i$ represents the *following* sampled state, i.e., a sample of $\xi(t_{i+1})$.

We employ the scenario-based optimization to estimate the matrices $M(\tau)$, equipped with guarantees of correctness. Drawing inspiration by the support vector regression (SVR) approach outlined in [15], we set up the problem

$$\min_{M(j),\gamma,\theta_i \geq 0} \quad \gamma + \rho \sum_{i=1}^{N} \theta_i$$
$$\text{s.t.} \quad \|\mathbf{X}'_i - M(\mathbf{Y}_i) \cdot \mathbf{X}_i\| - \gamma \leq \theta_i \quad (26)$$

where $\gamma$ is a program variable that defines a maximum prediction error, $\theta_i$ is a slack variable, and $\rho$ is a hyperparameter as per program (16). This setup encompasses noisy data (if $\gamma^* > 0$) or noiseless data (if $\gamma^* = 0$, or up to the solver's numerical precision). The program (26) provides the following probability guarantees.

***Theorem 3 (see [15], Th. 3):*** Let us consider a confidence parameter $\beta > 0$ and $\underline{\epsilon}$ and $\overline{\epsilon}$ defined as per Theorem 1. Given a new sample $(\mathbf{X}, \mathbf{Y}, \mathbf{X}')$, we have

$$\mathbb{P}^N[\, \underline{\epsilon}(s^*) \leq \mathbb{P}[\|\mathbf{X}' - M^*(\mathbf{Y}) \cdot \mathbf{X}\| > \gamma^*] \leq \overline{\epsilon}(s^*)\, ]$$
$$\geq 1 - \beta \quad (27)$$

where $M^*(j)\ \forall\, j$ are the matrices found by solving (26), and $s^*$ is the number of samples for which $\|\mathbf{X}' - M^*(j) \cdot \mathbf{X}\| > \gamma^*$ and $\mathbf{Y} = j$ is the IST associated with $\mathbf{X}$. □

Once the matrices $M^*(j)$ are found, we may follow the approach in [4], as the model-based abstraction can be built with the matrices $M(\tau)$ and the triggering matrix $Q$ [see (2)], which is assumed to be known.

Notice that Theorem 3 holds if we assume to know the IST $\mathbf{Y} = j$ associated with $\mathbf{X}$. This assumption is not restrictive, as we hypothesize to collect trajectories composed of samples $\xi(t_i)$, where the timing $t_i$ is an integral part of the data points. The intersample $\tau(\xi)$ can be recovered by simply computing $t_{i+1} - t_i$ for the samples of a trajectory: the IST represents the so-called *side information* of an event-based system, which is often neglected.

## V. Experimental Evaluation

We illustrate our approach considering 2-D, 3-D, and 4-D linear systems, along with 2-D and 3-D nonlinear systems. We compare the data-driven abstraction against the identified-model-based approach using noisy measurements.

TABLE I
PERFORMANCE OF THE SVM WITH VERONESE EMBEDDING (LEFT) AND GAUSSIAN KERNEL (RIGHT) FOR THE NOISY 2-D SYSTEM

| $\ell$ | $L$ | $s^*$ | $\overline{\epsilon}$ | Time | $s^*$ | $\overline{\epsilon}$ | Time |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 142 | 0.044 | 1 | 149 | 0.029 | 1 |
| 5 | 18 | 302 | 0.082 | 5 | 1120 | 0.206 | 10 |
| 10 | 33 | 467 | 0.119 | 58 | 3735 | 0.485 | 31 |

TABLE II
RESULTS OF THE SYSTEM IDENTIFICATION VIA SVR PROCEDURE FOR THE LINEAR (TOP) AND NONLINEAR (BOTTOM) SYSTEMS

| | $L$ | $s^*$ | $N$ | $\underline{\epsilon}$ | $\overline{\epsilon}$ | $\gamma^*$ | Time [s] |
|---|---|---|---|---|---|---|---|
| 2-D | 3 | 5 | 10k | 0 | 0.002 | $2.5{\cdot}10^{-5}$ | 3 |
| 3-D | 3 | 51 | 15k | 0.001 | 0.006 | $2{\cdot}10^{-5}$ | 12 |
| 4-D | 10 | 75 | 20k | 0.002 | 0.006 | $2.5{\cdot}10^{-5}$ | 17 |
| 2-D-A | 5 | 66 | 10k | 0.003 | 0.011 | $4.3{\cdot}10^{-5}$ | 37 |
| 3-D | 5 | 62 | 10k | 0.006 | 0.017 | $6.5{\cdot}10^{-5}$ | 30 |
| 2-D-B | 5 | 2 | 10k | 0 | 0.001 | $4.7{\cdot}10^{-5}$ | 26 |

### A. Veronese Embedding and SVM Formulations

We first study how the Veronese embedding influences the SVM performance. We collect $N = 10^4$ random samples $Z_i = (\mathbf{X}_i, \mathbf{Y}_i)$, where $\mathbf{X}_i$ are corrupted by zero-mean Gaussian noise with standard deviation $10^{-3}$, evaluated from the 2-D linear system in [4], where

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad K = [0 \quad -5] \quad (28)$$

with triggering condition $|\xi(t) - \hat{\xi}(t)|^2 > \sigma^2 |\xi(t)|^2$, where $\sigma = 0.1$. We obtain a dataset with three possible ISTs, i.e., $\mathbf{Y}_i \in \{1, 2, 3\}$. We fix the hyperparameters of the optimization program (16) as $\beta = 10^{-6}$ and $\rho = 10^3$. The train data are used to solve the optimization problem (16) with the Veronese embedding and using a chain of classifiers with RBF kernel (similarly to Section II-E). We collect the number of violated constraints $s^*$ and compute the corresponding $\overline{\epsilon}$ bound according to (20). Table I reports the number of labels $L$, the number of violated constraints $s^*$, its corresponding bound $\overline{\epsilon}$ and the computational time of the SVM algorithm, for different values of $\ell$. Notice that, as $\ell$ increases, we have a greater number of violated constraints $s^*$, which deteriorates $\overline{\epsilon}$. In the case of RBF kernel, $s^*$ is the sum of the violated constraints of $L$ optimization programs (denoted $s^*_j$ in Section II-E), and the reported $\overline{\epsilon}$ is the maximum among $\overline{\epsilon}_j$. The RBF kernel is faster than the Veronese embedding for a large value of $\ell$, but its probability bound $\overline{\epsilon}$ is significantly higher. While the chain of classifiers is a more general framework (it can be applied to linear or nonlinear models), it accumulates the probability bounds for all $L$ labels and return a much more conservative estimate.

### B. Data-Driven Traffic Abstraction

Next, we construct abstractions: considering several values of $\ell$, i.e., the length of the sequence of ISTs, we report the number of states $|\mathcal{X}|$ and transitions $|\mathcal{E}|$, the probability bound $\overline{\epsilon}$, and the abstraction precision $\hat{\delta}$ in Table III. The increase of $\ell$, together with an obvious increase in the number of states and transitions, provides a smaller $\hat{\delta}$ in view of the decrease of the abstraction's nondeterminism (the number of states approaches

TABLE III
DATA-DRIVEN AND MODEL-BASED ABSTRACTIONS FOR THE NOISY 2-D (TOP), 3-D (MIDDLE), AND 4-D (BOTTOM) LINEAR SYSTEMS

| | Data-driven | | | | | Model-based | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\ell$ | $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\overline{\epsilon}$ | $\hat{\delta}$ | Time | $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\hat{\delta}$ | Time |
| 1 | 3 | 9 | 0.044 | 2.0 | 1 | 3 | 9 | 2.0 | 8 |
| 5 | 18 | 24 | 0.082 | 2.0 | 8 | 18 | 24 | 2.0 | 6 |
| 10 | 33 | 38 | 0.119 | 1.0 | 69 | 34 | 39 | 0.97 | 6 |
| 1 | 3 | 9 | 0.039 | 2.0 | 3 | 3 | 9 | 2.0 | 17 |
| 2 | 7 | 17 | 0.060 | 2.0 | 8 | 7 | 17 | 2.0 | 14 |
| 3 | 13 | 26 | 0.070 | 2.0 | 16 | 15 | 33 | 2.0 | 15 |
| 4 | 20 | 34 | 0.085 | 1.5 | 23 | 28 | 54 | 2.0 | 63 |
| 1 | 10 | 100 | 0.020 | 9.0 | 25 | $5^{\dagger}$ | 25 | 9.0 | 22 |
| 2 | 43 | 218 | 0.025 | 3.49 | 80 | 13 | 39 | 2.30 | 20 |
| 3 | - | - | - | - | NI | - | - | - | TO |

TABLE IV
DATA-DRIVEN AND MODEL-BASED ABSTRACTIONS FOR THE 2-D HOMOGENEOUS (TOP), 3-D HOMOGENEOUS (MIDDLE), AND 2-D NONHOMOGENEOUS (BOTTOM) NONLINEAR MODELS

| | Data-driven | | | | | Model-based | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\ell$ | $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\overline{\epsilon}$ | $\hat{\delta}$ | Time | $\lvert\mathcal{X}\rvert$ | $\lvert\mathcal{E}\rvert$ | $\hat{\delta}$ | Time |
| 1 | 5 | 25 | 0.004 | 4.0 | 22 | 5 | 25 | 4.0 | 62 |
| 5 | 21 | 28 | 0.004 | 0.76 | 63 | - | - | - | TO |
| 10 | 41 | 48 | 0.005 | 0.39 | 135 | - | - | - | TO |
| 1 | 5 | 25 | 0.030 | 4.0 | 5 | - | - | - | TO |
| 2 | 14 | 40 | 0.031 | 4.0 | 15 | - | - | - | TO |
| 3 | 27 | 59 | 0.031 | 4.0 | 31 | - | - | - | TO |
| 1 | 5 | 25 | 0.008 | 4.0 | 9 | 5 | 25 | 4.0 | 43 |
| 5 | 21 | 28 | 0.009 | 0.76 | 45 | - | - | - | TO |
| 10 | 41 | 48 | 0.009 | 0.39 | 94 | - | - | - | TO |

the number of transitions). We finally challenge our procedure with 3-D and 4-D models—the explicit dynamics are reported in [4]. We apply our procedure for several values of $\ell$ and report the results in Table III. We highlight that the computational time of our procedure remains low for every test, despite the large number of labels. The addition of the noise reveals the possible downsides of the data-driven approach. With noisy data, the SVM algorithm suffers from the more complex dataset (which becomes *not* linearly separable): the program returns a numerical instability (denoted NI) for the 4-D system, $\ell = 3$—the same model takes roughly 2 min for the noiseless case, as shown in [18].

### C. Identified-Model-Based Approach

We challenge the identification procedure with samples corrupted by noise and further compare the identified-model-based approach against the data-driven traffic abstraction.

We report the results from the SVR technique (26) in terms of probability bounds $\underline{\epsilon}$ and $\overline{\epsilon}$ and the prediction cone $\gamma^*$ in Table II. Notice how the bounds $\underline{\epsilon}$ and $\overline{\epsilon}$ increase as $s^*$ increases—the effect is slightly mitigated by the increase of $N$ for the linear case. The probability bounds are attached to the matrices $M(\tau)$—we consider $\ell = 1$, as the matrices represent the one-step dynamics of models (1) and (2). The identified matrices give correct predictions for the next state up to $\gamma^*$. The model-based approach in [4] considers solely linear systems with perfect knowledge, i.e., with $\gamma^* = 0$, and it is out of the scope of this work to extend ETCetera to a noisy linear system. We thus employ the identified matrices in ETCetera without adding any noise-related or $\gamma^*$-related notion. To encompass noisy systems, one may employ iMDPs (see, e.g., [5]), or, in the case of bounded noise, one could follow a similar model-based abstraction, where the cones are *inflated*, and consider a nondeterministic mapping between the state $\xi(\cdot)$ and the IST $\tau$—as briefly conjectured in [4]. These approaches lead to more transitions and wider $\hat{\delta}$. Thus, results in Tables III and IV show a *best estimate* for $\hat{\delta}$ and the time computation. Similarly, the values of $\overline{\epsilon}$ in Table II represent a lower bound of the misclassification probability for the model-based procedure in Tables III and IV, as the identification procedure of an $\ell$-sequence entails selecting $\ell$ different ISTs.

Regarding the comparison between model-based and data-driven approaches, we underline that the model-based approach is generally faster than the data-driven for small systems, as the 2-D example in Table III, as observed in [18] for deterministic systems. However, time results in Tables III and IV do not include the time needed for the identification of $M(\tau)$ matrices, as they compare purely the abstraction building time. The data-driven approach is consistently faster on a 3-D system, but the probability upper bound considerably increases with the increase of $\ell$. This issue can be mitigated by adding more samples, yet this likely increases the computational time of the procedure.

We highlight that the noiseless 4-D system, for $\ell = 1$, is abstracted into a ten-state S$\ell$-CA. However, the model-based abstraction (built from the identified matrices via the SVR) solely has five states. This occurs because a small error in the estimation of matrices $M(\tau)$ propagates to matrices $N_i$ [see (8)], which are then used to compute the state-space partitions, providing the abstract states. The SVR-based method is thus not robust: a small change in $M(\tau)$ matrices can produce a significant modification of the S$\ell$-CA.

### D. Nonlinear Dynamics

The model-based technique to compute the infinite behavior of a PETC model is developed solely for linear models. As we aim at comparing the data-driven technique against a model-based approach, we assume to know the underlying model type (polynomial, in this case) and its order. We map the samples into a polynomial embedding—similarly to the Veronese embedding, but including all terms of order 1, 2, and so on—such that estimating the coefficients of the polynomial terms becomes a *linear* problem. Thanks to the polynomial embedding, our procedure follows the linear case. If also the embedding is unknown, the model-based technique cannot be employed to compute the sampling performance.

We show the effectiveness of our method considering two 2-D systems and a 3-D nonlinear system. We report our results in Table IV. The nonlinear models' abstractions show a similar pattern to the linear case: as $\ell$ increases, the number of labels increases, deteriorating the SVM performance (i.e., the computational time and the number of violated constraints, leading to a higher $\overline{\epsilon}$ bound). At the same time, the abstraction's precision ($\hat{\delta}$) improves, as higher $\ell$ provides more insight about the behavior of the concrete system. For the model-based case, the polynomial embedding augments the system dimensions: the 2-D and 3-D models, translated, respectively, with a third- and second-order polynomial basis, become both 10-D. The increase in system variables quickly makes the abstraction building computationally heavy, until the procedure times out in almost all the experiments.

## VI. Conclusion

In this article, we have presented a method to estimate the sampling performance of a PETC system with unknown dynamics by means of a data-driven abstraction. We build an $\ell$-complete abstraction to return bounds on the AIST with PAC guarantees, for both linear and nonlinear systems, in the presence of noisy data. Challenged against a model-based tool, we then prove that our data-driven approach is convenient (i.e., computationally faster) for 3-D (and larger) systems while providing tight probability guarantees. We show that the data-driven approach is faster for models with more than two variables, and it is more robust to perturbation than the model-based procedure: a small discrepancy in the matrices $M(\tau)$ can lead to a substantial difference in the resulting traffic abstraction. Future work includes the extension to the scenario approach to neural classifiers to overcome the limitations of SVM. We also aim to study different sampling of $\mathcal{X}$ and how these may affect the results in terms of probability guarantees. Finally, additional development on the ETCetera codebase is needed to account for imprecise predictions, i.e., in the presence of $\gamma^* > 0$ and probabilistic bounds $\bar{\epsilon}$.

## Appendix A
## Assumptions for Multiclass SVM

It is sufficient to show that the optimization program (16) satisfies the three technical assumptions—see Section III-A—to prove that a multiclass SVM problem is a valid instance of a scenario program. The first assumption trivially holds by the independence of the collected samples. Existence and uniqueness of the solution $(W_j^*, b_j^*, \theta_i^*)$, $j \in \{1, \dots, L\}$, $i \in \{1, \dots N\}$, trivially follow as per the binary SVM case [13], [15]. The nonaccumulation condition concerns the description of the hyperplanes separating the labels ($\ell$-sequences); in practical terms, it considers the case where all samples belong to a unique label ($\ell$-sequence).

*Nonaccumulation:* Recalling (15), we require the satisfaction of the condition

$$\mathbb{P}[\max_{j \neq \mathbf{Y}_i} \{(W_{\mathbf{Y}_i}^T - W_j^T)\mathbf{X}_i + (b_{\mathbf{Y}_i} - b_j) - \zeta\} = 0] = 0 \quad (29)$$

where the number of possible labels $j = 1, \dots, L$ depends on the number of labels witnessed from data. We distinguish between two cases: 1) if the dataset solely shows one label (which happens with nonzero probability) and 2) if the dataset shows two or more labels.

1) *Only one witnessed label:* In this case, our multiclass SVM algorithm degenerates: in particular, we set up only one hyperplane with parameters $(W_{\mathbf{Y}_i}, b_{\mathbf{Y}_i})$ for all samples. We set a simplified constraint as

$$g(\mathbf{X}_i) = \zeta - W_{\mathbf{Y}_i}\mathbf{X}_i - b_{\mathbf{Y}_i} \leq \theta_i \quad \forall\, i = 1, \dots, N. \quad (30)$$

This algorithmic setting is exactly the binary SVM: scenario-type guarantees can be obtained by using a *heated* (i.e., adding a supplementary probability distribution over the labels, see [15]) version of the problem.

2) *Two or more witnessed labels:* Let us focus on the case with two labels; the more general setting follows a similar argument. We set up two hyperplanes defined by $(W_1, b_1)$ and $(W_2, b_2)$. A solution for particularly simple datasets is represented by $(W_1, b_1) = (\mathbf{0}, 0)$. In this case, $W_2$ must be nonzero: if $W_2 = \mathbf{0}$ as well, the SVM solution could not be optimal as the classification decision [see (12)] reduces to

$$D(\mathbf{X}_i) = \arg\max_j \{W_j\mathbf{X}_i + b_j\} = \\ = \arg\max\{0, b_2\} = \begin{cases} \text{class } 2, & \text{if } b_2 > 0 \\ \text{class } 1, & \text{otherwise} \end{cases} \quad (31)$$

for every sample $\mathbf{X}_i$; therefore, every sample belonging to one of the two labels has a wrong classification, and such a solution is clearly not a minimum of program (16).
For $(W_1, b_1) = (\mathbf{0}, 0)$, the nonaccumulation specification can then be written as

$$\mathbb{P}[(W_1^T - W_2^T)\mathbf{X}_i + (b_1 - b_2) - \zeta = 0] = \\ \mathbb{P}[W_2^T\mathbf{X}_i = -(b_2 + \zeta)] = 0 \quad (32)$$

which is satisfied as $\mathbf{X}_i$ is a random variable—and so is $W_2^T\mathbf{X}_i$, for any nonzero $W_2$—hence, the probability of being equal to any value is trivially zero.

## Appendix B
## Discussion on Theorems 1 and 2

Appendix A proves that the multiclass SVM is a valid candidate for a scenario approach. Hence, we can leverage the discussion in [15] and claim that

$$\mathbb{P}^N\{\underline{\epsilon}(s^*) \leq R(\mathfrak{S}^*) \leq \bar{\epsilon}(s^*)\} \geq 1 - 3\beta \quad (33)$$

as outlined in Theorem 1. We remark that constraint violation in the SVM program (16) does not correspond to a misclassification, but rather marks that the difference between $h_{\mathbf{Y}_i}(\mathbf{X}_i)$—i.e., the function [see (11)] corresponding to the correct label evaluated at sample $\mathbf{X}_i$—and $h_j(\mathbf{X}_i)$—i.e., any other function $j \neq \mathbf{Y}_i$ evaluated at $\mathbf{X}_i$—is smaller than the threshold $\zeta$. A misclassification (left-hand-side term), on the other hand, corresponds to

$$h_{\mathbf{Y}_i}(\mathbf{X}_i) < h_j(\mathbf{X}_i) \implies h_{\mathbf{Y}_i}(\mathbf{X}_i) < h_j(\mathbf{X}_i) + \zeta \quad (34)$$

in view of $\zeta > 0$, which implies a constraint violation (right-hand-side term). This implication is strict, and misclassification is more rare than constraint violation: therefore, Theorem 1 can be used to only upper bound the probability of misclassification, as per Theorem 2.

## Appendix C
## Alternative Multiclass SVM Formulations

The approach presented in Section II-D uses the Crammer–Singer [26] formulation for a multiclass SVM problem, which allows us to express the hyperplane synthesis in a single optimization program. The scenario theory can also be applied to the other two common multiclass formulations, i.e., *one-vs-all* and *one-vs-one*.

1) *One-vs-all:* This problem is similar to the Crammer–Singer expression presented in the main article: it also synthesizes $L$ hyperplanes for $L$ classes in one single optimization program and the label decision involves $\arg\max$ among all constraints. Formally, it can be written as

$$\begin{aligned}\min_{W,b,\xi_i\geq 0} \quad & \sum_{i=1}^{L}||W_i||^2+\rho\sum_{i=1}^{N}\theta_i \\ \text{s.t.} \quad & W_j\mathbf{X}_i+b_j\geq\zeta-\theta_i, \qquad \text{if } \mathbf{Y}_i=j \\ & W_j\mathbf{X}_i+b_j\leq-\zeta+\theta_i, \qquad \text{if } \mathbf{Y}_i\neq j. \end{aligned} \tag{35}$$

Recall that the scenario approach requires one constraint per sample; hence, we rewrite the constraints as

$$\begin{aligned}&\max\{-W_{\mathbf{Y}_i}\mathbf{X}_i-b_{\mathbf{Y}_i}+\zeta,\ W_j\mathbf{X}_i+b_j+\zeta\}\leq\theta_i \\ &\forall\, i,\ \text{for } j\neq\mathbf{Y}_i.\end{aligned} \tag{36}$$

Existence, uniqueness, and nonaccumulation are satisfied similarly to the Crammer–Singer formulation, and thus, the results of Theorem 1 can be applied also for the one-vs-all case.

2) *One-vs-one:* The one-vs-one formulation extends a binary classifier in a hierarchical sense, in order to accommodate for multiple classes. In practical terms, the one-vs-one approach trains $M:=L(L-1)/2$ classifiers in separate optimization programs, one for each couple of labels. We may think of this as casting a vote among all possible couple of labels, and the final decision on the predicted label depends on which class obtains more votes.

This methodology tacitly imposes $M$ optimization programs; hence, we must account for them separately in the scenario approach, i.e., during the computation of bound $\overline{\epsilon}$. When considering the classifier between the $i$th and $j$th classes, we solve the following problem:

$$\begin{aligned}\min_{w^{ij},b^{ij},\theta^{ij}\geq 0} \quad & \frac{1}{2}(w^{ij})^Tw^{ij}+\rho\sum_t\theta_t^{ij} \\ \text{s.t.} \quad & y_t(w^{ij}\mathbf{X}_t+b^{ij})+\zeta\leq\theta_t^{ij}\end{aligned} \tag{37}$$

if we recast $y_t\in\{-1,+1\}$. The existence, uniqueness of the solution, and nonaccumulation assumptions of the overall classifier are satisfied as per [15], as we are simply setting up $M$ binary classification procedures.

Notice that every $m$th optimization program considers a different number of samples: the binary classifier separating classes $i$ and $j$ uses solely the samples belonging to the corresponding classes, for a total of $N_m=N_i+N_j$ samples (where $N_i$ is the number of samples belonging to class $i$). The number of samples per classifier $N_i+N_j$ is much smaller than the total number of samples $N$: this degrades the probability bounds, as $\overline{\epsilon}$ is inversely proportional to $N$.

Furthermore, every binary classifier returns a different number of violated constraints, denoted $s_m^*$; thus, for each classifier, we obtain the probability bound $\overline{\epsilon}(s_m^*,N_m,\beta)$—denoted $\overline{\epsilon}_m$ for brevity. This probability bound holds *for the single classifier*, while the overall classification procedure assigns a label to each sample according to a voting procedure.

Each label is taken into account by $(L-1)$ classifiers and ideally should receive $(L-1)$ votes (i.e., the label wins all the votes where it is present). If this occurs, the label surely wins the ballot—as no other label can receive more than $(L-2)$ votes—and is assigned to the given sample. Other voting results are possible in order to win the ballot, but for simplicity we will focus on the requirement that a label must win all possible votes to be the final classifier decision.

Every single classifier returns the correct vote with probability (not smaller than) $(1-\overline{\epsilon}_m)$; hence, the event of correct vote for a label $l$ from all possible classifiers occurs with probability

$$\varepsilon_l=\prod_{i\in V(l)}(1-\overline{\epsilon}_i) \tag{38}$$

where $V(l)$ denotes the voting pool for label $l$, i.e., the classifiers that consider label $l$. The quantity $\varepsilon_l$ is a product of exactly $(L-1)$ terms: if all classifiers share the same probability bound $\overline{\epsilon}_i$, then $\varepsilon_l=(1-\overline{\epsilon}_i)^{(L-1)}$. In general, however, each label activates classifiers with different probability bounds; hence, each $\varepsilon_l$ takes different values; we shall denote the minimum among these as

$$\underline{\varepsilon}:=\min_{l=1,\dots,L}\varepsilon_l. \tag{39}$$

Following the one-vs-one procedure, the result of Theorem 1 may be revised as follows:

$$\mathbb{P}^N[R(\mathfrak{S}^*)\leq 1-\underline{\varepsilon}]\geq 1-3\beta. \tag{40}$$

*Example:* Let us assume a dataset composed of 1000 samples belonging to three labels, namely, $A$, $B$, and $C$. The sample set is not balanced; the number of samples per label is $N_A=600$, $N_B=300$, and $N_C=100$. The one-vs-one procedure creates three classifiers, one for each couple of labels: let us denote them $c_{AB}$, $c_{AC}$, and $c_{BC}$. Let us assume that the classifiers return a number of constraint violations as $s_{AB}^*=7$, $s_{AC}^*=3$, and $s_{BC}^*=2$. We compute the probability bound $\overline{\epsilon}_{i,j}$ for every classifier, setting the confidence $\beta=10^{-3}$, and obtain the values

$$\begin{aligned}\overline{\epsilon}_{AB}&=\overline{\epsilon}(s_{AB}^*,N_A+N_B,\beta)=0.027\\ \overline{\epsilon}_{AC}&=\overline{\epsilon}(s_{AC}^*,N_A+N_C,\beta)=0.024\\ \overline{\epsilon}_{BC}&=\overline{\epsilon}(s_{BC}^*,N_B+N_C,\beta)=0.036.\end{aligned}$$

The probability of a correct classification of label $A$ is given by the probability of a correct classification of both the classifiers $c_{AB}$ and $c_{AC}$; similarly for labels $B$ and $C$. This results in

$$\begin{aligned}\varepsilon_A&=(1-\overline{\epsilon}_{AB})(1-\overline{\epsilon}_{AC})=0.949\\ \varepsilon_B&=(1-\overline{\epsilon}_{AB})(1-\overline{\epsilon}_{BC})=0.937\\ \varepsilon_C&=(1-\overline{\epsilon}_{AC})(1-\overline{\epsilon}_{BC})=0.940.\end{aligned} \tag{41}$$

We then consider the smallest among them, $\underline{\varepsilon}=0.937$, and obtain the conservative estimate $1-\underline{\varepsilon}=0.063$ as

$$\mathbb{P}^N[R(\mathfrak{S}^*)\leq 0.063]\geq 0.997. \tag{42}$$

## REFERENCES


[1] K. J. Astrom and B. M. Bernhardsson, "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," in *Proc. 41st IEEE Conf. Decis. Control*, 2002, vol. 2, pp. 2011–2016.

[2] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Trans. Autom. Control*, vol. 52, no. 9, pp. 1680–1685, Sep. 2007.

[3] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *Proc. IEEE 51st IEEE Conf. Decis. Control*, 2012, pp. 3270–3285.

[4] G. de A. Gleizer and M. Mazo Jr., "Computing the sampling performance of event-triggered control," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–7.

[5] G. Delimpaltadakis and M. Mazo, "Region-based self-triggered control for perturbed and uncertain nonlinear systems," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 2, pp. 757–768, Jun. 2021.

[6] G. Delimpaltadakis, G. de Albuquerque Gleizer, I. van Straalen, and M. Mazo Jr., "ETCetera: Beyond event-triggered control," in *Proc. 25th Int. Conf. Hybrid Syst.: Comput. Control*, 2022, pp. 1–11.

[7] D. Haussler, "Probably approximately correct learning," in Proc. 8th Nat. Conf. Artif. Intell., 1990, pp. 1101–1108.

[8] T. Badings, M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, and U. Topcu, "Scenario-based verification of uncertain MDPs," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2020, pp. 287–305.

[9] T. S. Badings et al., "Sampling-based robust control of autonomous systems with non-Gaussian noise," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 9669–9678.

[10] A. Devonport, A. Saoud, and M. Arcak, "Symbolic abstractions from data: A PAC learning approach," in *Proc. 2021 60th IEEE Conf. Decis. Control*, 2021, pp. 599–604.

[11] R. Coppola, A. Peruffo, and M. Mazo, "Data-driven abstractions for verification of linear systems," *IEEE Control Syst. Lett.*, vol. 7, pp. 2737–2742, 2023.

[12] M. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM J. Optim.*, vol. 19, no. 3, pp. 1211–1230, 2008.

[13] M. C. Campi, S. Garatti, and F. A. Ramponi, "A general scenario theory for nonconvex optimization and decision making," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4067–4078, Dec. 2018.

[14] S. Garatti and M. C. Campi, "Risk and complexity in scenario optimization," *Math. Program.*, vol. 191, pp. 243–279, 2019.

[15] M. C. Campi and S. Garatti, "Scenario optimization with relaxation: A new tool for design and application to machine learning problems," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 2463–2468.

[16] J. Berberich, S. Wildhagen, M. Hertneck, and F. Allgöwer, "Data-driven analysis and control of continuous-time systems under aperiodic sampling," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 210–215, 2021.

[17] S. Wildhagen, J. Berberich, M. Hertneck, and F. Allgöwer, "Data-driven analysis and controller design for discrete-time systems under aperiodic sampling," *IEEE Trans. Autom. Contr.*, vol. 68, no. 6, pp. 3210–3225, 2022.

[18] A. Peruffo and M. Mazo, "Data-driven abstractions with probabilistic guarantees for linear PETC systems," *IEEE Control Syst. Lett.*, vol. 7, pp. 115–120, 2023.

[19] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel, "Periodic event-triggered control for linear systems," *IEEE Trans. Autom. Control*, vol. 58, no. 4, pp. 847–861, Apr. 2013.

[20] R. Postoyan, R. G. Sanfelice, and W. P. M. H. Heemels, "Explaining the "mystery" of periodicity in inter-transmission times in two-dimensional event-triggered controlled system," *IEEE Trans. Autom. Control*, vol. 68, no. 2, pp. 912–927, Feb. 2023, doi: 10.1109/TAC.2022.3147009.

[21] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. New York, NY, USA: Springer, 2009.

[22] K. Chatterjee, L. Doyen, and T. A. Henzinger, "Quantitative languages," *ACM Trans. Comput. Log.*, vol. 11, no. 4, pp. 1–38, 2010.

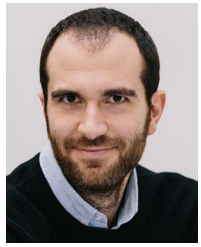
[23] R. M. Karp, "A characterization of the minimum cycle mean in a digraph," *Discrete Math.*, vol. 23, no. 3, pp. 309–311, 1978.

[24] J. Weston and C. Watkins, "Multi-class support vector machines," Dept. Comput. Sci., Roy. Holloway, Univ. London, Egham, U.K., Tech. Rep. CSD-TR-98-04, 1998.

[25] J. Harris, *Algebraic Geometry: A First Course*, vol. 133. New York, NY, USA: Springer, 2013.

[26] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, 2001.

[27] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, 2011.

**Andrea Peruffo** (Member, IEEE) received the Laurea degree in information engineering and the M.S. degree in automation engineering from the University of Padova, Padova, Italy, in 2012 and 2015, respectively, and the D.Phil. degree in computer science from the University of Oxford, Oxford, U.K., in 2021.

In 2016, he was a Research Engineer with Inria Rennes, Rennes, France. In 2020, he visited the Erato MMSD Group, National Institute of Informatics, Tokyo, Japan. He is currently a Postdoctoral Researcher with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. His research interests include hybrid systems, formal verification, and control theory.

**Manuel Mazo Jr.** (Senior Member, IEEE) received the Telecommunications Engineering degree from the Polytechnic University of Madrid, Madrid, Spain, in 2003, and the Electrical Engineering degree (awarded under a TIME double-degree program) from the Royal Institute of Technology, Stockholm, Sweden, in 2003, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, CA, USA, in 2007 and 2010, respectively.

He spent two years as a Postdoctoral Researcher with INCAS3, Assen, The Netherlands, and the University of Groningen, Groningen, The Netherlands. He is currently an Associate Professor with the Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands.