Delft University of Technology

Data-driven Abstractions with Probabilistic Guarantees for Linear PETC Systems

Peruffo, Andrea; Mazo, Manuel

DOI
10.1109/LCSYS.2022.3186187

Publication date
2022

Document Version
Final published version

Published in
IEEE Control Systems Letters

Citation (APA)
Peruffo, A., & Mazo, M. (2022). Data-driven Abstractions with Probabilistic Guarantees for Linear PETC Systems. *IEEE Control Systems Letters*, *7*, 115-120. https://doi.org/10.1109/LCSYS.2022.3186187

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher
is the copyright holder of this work and the author uses the
Dutch legislation to make this work public.

Data-Driven Abstractions With Probabilistic Guarantees for Linear PETC Systems

Andrea Peruffo, *Fellow, IEEE*, and Manuel Mazo, Jr., *Fellow, IEEE*

***Abstract*—We employ the scenario approach to compute probably approximately correct (PAC) bounds on the average inter-sample time (AIST) generated by an unknown PETC system, based on a finite number of samples. We extend the scenario optimisation to multiclass SVM algorithms in order to construct a PAC map between the concrete state-space and the inter-sample times. We then build a traffic model applying an ℓ-complete relation and find, in the underlying graph, the cycles of minimum and maximum average weight: these provide lower and upper bounds on the AIST. Numerical benchmarks show the practical applicability of our method, which is compared against model-based state-of-the-art tools.**

***Index Terms*—Automata, discrete event systems, statistical learning.**

I. INTRODUCTION

IN THE last decades, thanks to the increasing digitalisation and use of communication networks, control systems have faced several new challenges. Arguably, one of the most compelling tasks is ensuring the stability of an interconnection between an analog plant and a digital controller, possibly in spite of imperfect communication means. The interface between a (digital) controller and an (analog) plant is typically implemented with a periodic sampling of the plant, whose measurements are transmitted to the controller, which computes the actions in order to optimise a performance cost. The sampling period itself represents a tradeoff between control performance and energy consumption.

Event-triggered control (ETC) is a paradigm that tackles this issue, adjusting the sampling according to the satisfaction of a condition, whilst maintaining the stability guarantees. This notion has originally been developed in [1], and further notably developed in [2], where guarantees on the closed loop performance are ensured. A practical implementation procedure of this scheme is the periodic ETC (PETC): the stability condition is periodically checked, albeit the measurement is transmitted solely when the condition is verified – see, e.g., [3], [4]. Whilst the advantage of this approach is apparent, formally and quantitatively *measuring* its performance has been tackled only recently.

The use of formal abstractions for PETC models has been preparatory in this sense, see, e.g., [5], [6]. More recently, the work [4] provides a characterisation in terms of traffic abstractions: a finite-state automaton considering the inter-sample times (ISTs) sequences of a PETC system. The procedure is applied in [3], shifting the focus to ℓ consecutive inter-sample times: this offers a conservative estimate of the long-term performance of a PETC system. Further, in [7] the authors build a traffic abstraction in order to compute the minimum average inter-sample time (AIST) of a PETC system, a metric that can be translated directly to the expected network load or resource utilization. Computing the sampling performance allows us to evaluate the controller's performance and to compare different (P)ETC implementations.

ETCetera, a state-of-the-art tool to compute the sampling performance of ETC systems is presented in [8].

We build upon this literature, aiming at constructing finite-state abstraction of a linear PETC system, with a crucial difference. In this letter, we assume to solely collect samples from the underlying concrete system. Nevertheless, we construct a data-driven map between the concrete state-space and the inter-sample times with probably approximately correct (PAC) guarantees, which acts as the foundation for a traffic abstraction to compute reliable bounds on the AIST of the unknown system. Data-driven abstractions have recently gained interest, e.g., in [9], [10], where the authors provide bounds on the transition probabilities, and in [11] where a behavioural relationship is defined.

Since the concrete model is unknown, building the relationship between the state-space and the corresponding ISTs is a non-trivial challenge. We provide a map based on a finite number of samples (often called scenarios) which we assume can be computed via simulations. We adapt the scenario approach that provides probabilistic guarantees by solving a convex optimisation program [12]–[14]. This map, computed by collecting ℓ-long sequences of ISTs, partitions the underlying state space. Each partition corresponds to an abstract state of the traffic model, whose transitions are governed by the so-called domino rule. The traffic abstraction offers the upper and lower bounds on the AIST; we may refine this procedure by increasing ℓ until the upper and lower bounds reach a desired precision. The

Manuscript received 21 March 2022; revised 24 May 2022; accepted 9 June 2022. Date of publication 24 June 2022; date of current version 11 July 2022. This work was supported by the European Research Council through the SENTIENT Project (ERC-2017-STG) under Grant 755953. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Andrea Peruffo.)*

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

Digital Object Identifier 10.1109/LCSYS.2022.3186187

2475-1456 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

traffic abstraction derived from this map, equipped with PAC bounds, is employed to compute the sampling performance of the underlying PETC system.

Contributions: In this letter, we elaborate an extension to [14] as we tailor the scenario approach to multiclass SVM (support vector machine) algorithms. We employ this result to build a data-driven abstraction of an unknown linear PETC system with probabilistic guarantees of correctness. We aim at estimating its resource consumption by identifying the possible (infinite) sequences of inter-sample times. To this end, we provide probably correct bounds that tightly approximate the desired consumption cost. Finally, we compare our procedure against the model-based tool ETCetera [8].

II. FINITE ABSTRACTION OF PETC VIA TS

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

$$\dot{\xi}(t) = A\xi(t) + B\nu(t), \qquad \nu(t) = K\hat{\xi}(t), \tag{1}$$

where $\xi(t) \in \mathbb{R}^{n_x}$ is the plant's state with initial value $\xi_0 = \xi(0)$, $\hat{\xi}(t) \in \mathbb{R}^{n_x}$ is the state measurement available to the controller, $\nu(t) \in \mathbb{R}^{n_u}$ is the control input, n_x and n_u are the state-space and input-space dimensions, respectively, and A, B, K are matrices of appropriate dimensions. The measurements are sent to the controller 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 ETC, a triggering condition determines the sequence of times t_i. In the case of PETC, this condition is checked only periodically, with a fundamental checking period h.

We typically consider the family of quadratic triggering conditions [15] where we additionally set a maximum inter-sampling time $\overline{\kappa}$, known as the *heartbeat* of the system. Formally, we define the $(i+1)$-th triggering time as

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

where $k \in \mathbb{N}$ and $Q \in \mathbb{S}^{2n_x}$ is the designed triggering matrix, and $\overline{\kappa}$ is the chosen maximum inter-sample time. Every run of system (1), starting from initial condition $\xi(0)$, generates an infinite sequence of samples $\{\xi(t_i)\}$ and of inter-sample times $\{\tau(\xi(t_i))\}$ – for brevity denoted $\{\tau_i\}$. In the following, we solely consider PETC settings hence $\tau \in h\cdot\{1, 2, \dots \overline{\kappa}\}$; for simplicity and without loss of generality, we consider $h = 1$.

A typical metric of interest is the *average inter-sample time* (AIST), defined for every initial state $\xi(0)$ as

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

Notice that we use the liminf instead of lim in case the regular limit does not exist [16], making the AIST a well-defined metric. Additionally we define the *smallest* and the *largest* average inter-sample time (SAIST and LAIST, respectively) as

$$\mathrm{SAIST} := \inf_{\xi_0\in\mathbb{R}^{n_x}} \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau(\xi(t_i)), \tag{4}$$

$$\mathrm{LAIST} := \sup_{\xi_0\in\mathbb{R}^{n_x}} \limsup_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \tau(\xi(t_i)). \tag{5}$$

Both the SAIST and the LAIST are crucial metrics to assess the performance (in terms of resources consumption or band utilisation) of an ETC implementation. However, the present expressions (4), (5) require the search for a state ξ_0 over $\mathbb{R}^{n_x}$ and the choice of a sufficiently large n, which are tasks computationally hard (if at all possible). To overcome this impediment, we approach this problem exploiting the (finite-state) abstractions theory.

A. Abstractions and Transition Systems

The abstraction procedure allows the analysis of large (even infinite) models. We usually consider a *concrete* model, which is treated as a ground truth; this is transformed, simplified, and adapted to accommodate the analysis of its behavior. Several kinds of abstractions are available for different purposes: we employ a finite-state abstraction in the form of a (weighted) transition system (WTS).

Definition 1 (Weighted Transition System (Adapted From [6], [17])): A transition system $\mathcal{S}$ is a tuple $(\mathcal{X}, \mathcal{X}_0, \mathcal{E}, \mathcal{Y}, \mathcal{H}, \gamma)$ where:

- $\mathcal{X}$ is the set of states,
- $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states,
- $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of edges, or transitions,
- $\mathcal{Y}$ is the set of outputs, and
- $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ is the output map,
- $\gamma : \mathcal{E} \to \mathbb{Q}$ is the weight function.

The original definition also includes the action set $\mathcal{U}$, which is here omitted since we solely consider autonomous systems; we further consider finite-state systems, where the cardinality of $\mathcal{X}$ is finite. We tacitly consider *non-blocking* transition systems, i.e., automata where every state is equipped with at least one outgoing transition.

Let us define $r = \mathrm{x}_0\mathrm{x}_1\mathrm{x}_2\dots$ an infinite internal behavior, or *run* of $\mathcal{S}$ if $\mathrm{x}_0 \in \mathcal{X}_0$ and $(\mathrm{x}_i, \mathrm{x}_{i+1}) \in \mathcal{E}$ for all $i \in \mathbb{N}$, and, with slight abuse of notation, $\mathcal{H}(r) = \mathrm{y}_0\mathrm{y}_1\mathrm{y}_2\dots$ its corresponding *external* behavior, or trace, if $\mathcal{H}(\mathrm{x}_i) = \mathrm{y}_i$ for all $i \in \mathbb{N}$. Similarly, we define $\gamma(r) = \nu_0\nu_1\dots$ the sequence of weights from run r, where $\gamma(\mathrm{x}_i, \mathrm{x}_{i+1}) = \nu_i$. For simplicity, we consider the weight function equal to the outbound state output, i.e., $\gamma(\mathrm{x}, \mathrm{x}') = \mathcal{H}(\mathrm{x})$. Let us further consider a value function, i.e., a function mapping an infinite sequence of weights to a (possibly finite) value, as

$$\mathrm{LimAvg}(\gamma(r)) := \liminf_{n\to\infty} \frac{1}{n+1} \sum_{i=0}^{n} \nu_i. \tag{6}$$

A weighted automaton equipped with the LimAvg value function is called a LimAvg-automaton [17].

Closely related to this metric, we can define

$$\overline{V}(\mathcal{S}) := \sup\{\mathrm{LimAvg}(\gamma(r)) \mid r \text{ is a run of } \mathcal{S}\}, \tag{7}$$

$$\underline{V}(\mathcal{S}) := \inf\{\mathrm{LimAvg}(\gamma(r)) \mid r \text{ is a run of } \mathcal{S}\}. \tag{8}$$

Remarkably, [17], [18] show that we can recover from a WTS the value of $\underline{V}(\mathcal{S})$ and $\overline{V}(\mathcal{S})$ in polynomial time. The value function in (7) evaluates to the *smallest average cycle* (SAC) of

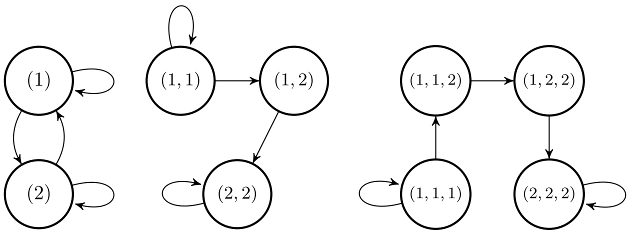

Fig. 1. Sℓ-CA with $\tau_i = \{1, 2\}$ and ℓ=1 (left), $\ell = 2$ (center), $\ell = 3$ (right).

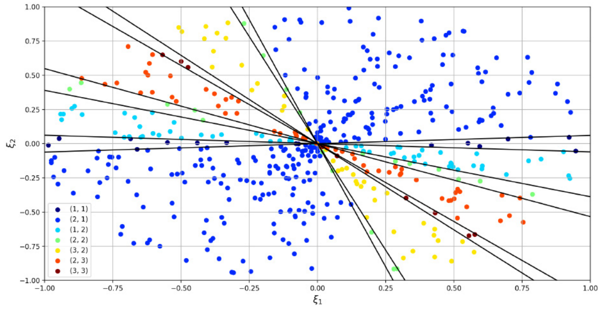

Fig. 2. IST regions are unions of (diametrically opposite) cones: black lines denote the regions' boundaries, depicted along with 1000 samples in different colors, for $\ell = 2$.

the graph underlying the transition system. In many contexts, the performance of systems as the (minimum) average resource consumption is modeled as the minimum cycle mean problem. Analogously, we may be interested in the *largest* average cycle (LAC) to analyse the maximum average resource consumption.

We ideally would abstract a PETC system as a WTS. However, the abstraction requires the knowledge on the internal behaviour of the system, i.e., the states ξ and their dynamics. We then introduce the notion of *behavioural inclusion*: the abstraction is oblivious to the internal behaviour of a system, but we require that all traces observed in the concrete system are also observed in the abstraction. In practical terms, the ISTs of any trajectory of the concrete PETC system must overlap the output of a run of the weighted TS.

A natural way of building a behavioural inclusion abstraction is by mapping each possible output symbol τ_i to an abstract state x_i. We may elaborate this intuition through a so-called ℓ-complete model:

Definition 2 ((Strongest) ℓ-Complete Abstraction [7]): Let $\mathcal{S} := (\mathcal{X}, \mathcal{X}_0, \mathcal{E}, \mathcal{Y}, \mathcal{H})$ be a transition system as per Definition 1, and let $\mathcal{X}_\ell \subseteq \mathcal{Y}^\ell$ be the set of all ℓ-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) ℓ-complete abstraction (Sℓ-CA) of $\mathcal{S}$, where

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

where we denote $B_\ell(\mathcal{S})$ all the possible external traces of system $\mathcal{S}$ and $\mathcal{Y}^\ell$ is the cartesian product $\mathcal{Y} \times \ldots \times \mathcal{Y}$ repeated ℓ times. The intuition behind the Sℓ-CA is to encode each state as an ℓ-long external trace. Notice that the transitions follow the so-called "domino rule": e.g., let us assume $\ell = 3$ and let us observe a trace *abc* (this trace corresponds to a single abstract state); the next ℓ-trace must begin with *bc*. Therefore, a transition from *abc* can lead to, e.g., *bca*, *bcb*, *bcc*. Finally, the output of a state is its first element: state *abc* has output $\mathcal{H}(abc) = a$.

The Sℓ-CA translation is particularly useful when little or no information is available about the concrete system, as only the external behaviour, i.e., the sequences of τ_i, is employed to build the abstraction. In order to compute the AIST, which is a function of $\xi(0)$, we shall need a map, denoted $\mathcal{T} : \mathbb{R}^{n_x} \to \mathcal{Y}^\ell$, that relates concrete states to inter-sample times in order to compute the initial abstract state x_0. Further, the Sℓ-CA is in general non-deterministic: we cannot evaluate exactly the LimAvg (as every run starting from x_0 branches into several possible runs), thus we compute the SAC and LAC (see (7)) and use them as upper and lower approximations of the LimAvg, i.e., the AIST. Figure 1 shows three examples of Sℓ-CA, where $\tau_i = \{1, 2\}$ and $\ell = 1, 2, 3$. Let us consider $\ell = 3$ and $\mathrm{x}_0 = (1, 1, 1)$ as initial state. The SAC and LAC correspond to the self-loop on states $(1, 1, 1)$ and $(2, 2, 2)$: the value of these cycles are 1 and 2, respectively. Therefore, the AIST is within $[1, 2]$.

The triggering condition (see (2)) allows solely a *subset* of all possible ℓ-sequences of ISTs. A model-based method can exploit the knowledge of the concrete system to check which sequences of length ℓ are admissible and construct a tailored TS [4], [7].

In the following, we employ a scenario approach to obtain the map $\mathcal{T}$ with guarantees of correctness and carefully tailor the abstract state space without any knowledge of the underlying concrete dynamics.

III. DATA-DRIVEN ABSTRACTIONS WITH GUARANTEES

Naturally, full knowledge of the concrete system is useful to compute the mapping $\mathcal{T}$ between the states $\{\xi(t_i)\}$ and the output traces $\{\tau_i\}$. However, in many applications this knowledge may be unavailable, unreliable, or simply expensive to obtain. To overcome this impediment, we employ the scenario approach [12].

Let us assume we collect N samples $Z_i = (\mathbf{X}_i, \mathbf{Y}_i)$, $i = 1, \ldots N$, where $\mathbf{X}_i$ represents the i-th sample of the concrete system state and $\mathbf{Y}_i$ represents an ℓ-sequence of ISTs. Since $\mathbf{Y}_i$ have finite cardinality, we may associate them with *classes* or *labels* from a machine learning viewpoint. From now on, we consider $\{\mathbf{Y}_i\}$ belonging to L different classes, i.e., $\mathbf{Y}_i \in \{1, \ldots L\}$.

For systems (1)-(2), the regions of the state space that correspond to ℓ-sequences of ISTs are unions of cones [3], [4], [7], as depicted in Fig. 2. Intuitively, cones arise from the quadratic condition (2), which are linearly transformed by the matrices in (1). We shall then use this intuition to partition the state space and construct the abstract states. This allows to reduce the number of abstracted states in comparison to an agnostic gridding procedure.

A. Linear Separability With Veronese Embedding

We draw inspiration from the SVM literature, where hyperplanes are constructed in order to split linearly separable datasets. Typically, when the samples are not linearly separable, we employ the kernel trick [19] to embed the data into a higher-dimensional space, where a linear separation is possible. A model given by (1)-(2) separates the state space into (unions of) cones: hence, a conic transformation to the samples $\mathbf{X}_i$ grants a linearly separable dataset. This embedding is known as the Veronese map [20] of order 2, $\mathcal{V}_2 : \mathbb{R}^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$, defined by

$$\mathcal{V}_2(x) := [x_1^2 \quad x_1 x_2 \quad x_2^2 \ \dots \ x_n^2], \tag{9}$$

where x_j indicates the j-th component of vector x. This embedding represents a tailored kernel trick for the problem at hand. Further, the conic partitions fulfil an *encapsulation* constraint: every point of the concrete system ξ, whose next IST is j, satisfies

$$\xi^T N_i \xi < 0 \text{ for } i < j, \text{ and } \xi^T N_j \xi > 0, \tag{10}$$

where the set of matrices N_i is a function of A, B, K and R (see (1)-(2)). The exact formulation of N_i can be found in, e.g., [4], [7]. Intuitively, the N_i inequalities indicate a violation of the i-th triggering condition. For instance, if $\xi^T N_1 \xi < 0$, the triggering condition is not violated after 1 inter-sample times; conversely, if $\xi^T N_2 \xi > 0$ the triggering condition is violated after 2 ISTs, provided that it is not violated after 1 IST (i.e., $\xi^T N_1 \xi < 0$ must hold).

We map conditions (10) with the Veronese embedding and obtain a set of linear constraints

$$W_i \cdot \mathcal{V}_2(\xi) < 0 \text{ for } i < j, \text{ and } W_j \cdot \mathcal{V}_2(\xi) > 0, \tag{11}$$

where W_i shall be recovered from the samples Z_i. We now outline the scenario theory for a (more general) SVM multiclass problem: the encapsulation constraints (11) can be written as a particular case of the SVM framework. We denote *conic multiclass SVM* (CM-SVM) the SVM algorithm with Veronese embedding and encapsulation constraints.

B. Sample-Based Partitioning via Classification

Building upon [14], where binary SVM with scenario guarantees is presented, we extend this approach to multiclass SVM algorithms. This procedure provides a classifier with guarantees of correctness, which we denote as the map $\mathcal{T}$. Multiclass classification belongs to two families [21]: the *one-vs-one* reduction, where the multi-class problem is split into multiple binary classification problems, and the *one-vs-all* approach, which defines one hyperplane per class. Notably, the one-vs-all approach can be written in one single optimisation program: for clarity, we select the Crammer-Singer formulation [22]. The extension to other formulations proceeds similarly to the following discussion.

The one-vs-all SVM algorithm consists of one convex problem considering L (one per class) hyperplanes, defined

$$f_j(\mathbf{X}_i) := W_j \mathbf{X}_i + b_j, \ \text{ for } j \in \{1, \dots, L\}, \tag{12}$$

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

$$D(\mathbf{X}_i) := \arg\max_j \{W_j \mathbf{X}_i + b_j\}. \tag{13}$$

For every sample $\mathbf{X}_i$, the corresponding hyperplane is denoted as $(W_{\mathbf{Y}_i}, b_{\mathbf{Y}_i})$, i.e., the $\mathbf{Y}_i$-th hyperplane. For a correct classification, the value of hyperplane $W_{\mathbf{Y}_i}\mathbf{X}_i + b_{\mathbf{Y}_i}$ ought to be greater than all others:

$$\zeta - (W_{\mathbf{Y}_i} - W_j)\mathbf{X}_i + (b_{\mathbf{Y}_i} - b_j) \le 0, \quad \forall j \neq \mathbf{Y}_i, \ \forall i, \tag{14}$$

where the hyper-parameter $\zeta > 0$ is added for numerical stability. Let us define the function $g(\mathbf{X}_i, \mathbf{Y}_i)$ as

$$g(\mathbf{X}_i, \mathbf{Y}_i) := \max_{j \neq y_i} \big\{ \zeta - (W_{\mathbf{Y}_i} - W_j)\mathbf{X}_i - (b_{\mathbf{Y}_i} - b_j) \big\}. \tag{15}$$

Constraints $g(\mathbf{X}_i, \mathbf{Y}_i) \le 0$ can be satisfied only if the dataset is linearly separable: to further accommodate for more general cases, we employ slack variables $\theta_i \ge 0$, $i = 1, \dots, N$, impose $g(\mathbf{X}_i, \mathbf{Y}_i) \le \theta_i$ and minimise the sum of θ_i, as outlined in, e.g., [19]. An L-class classifier can be obtained by solving the following convex program:

$$\begin{aligned} \min_{W, b, \theta_i \ge 0} \ & \sum_{j=1}^{L} ||W_j||^2 + \rho \sum_{i=1}^{N} \theta_i, \\ \text{s.t.} \ & g(\mathbf{X}_i, \mathbf{Y}_i) \le \theta_i, \quad i = 1, \dots, N, \end{aligned} \tag{16}$$

where $\rho > 0$ is a hyper-parameter balancing the trade-off between the correctness of the algorithm (the number of positive θ_i) and its cost (the norm of matrices W_j).

Let us now introduce a quantitative measure to evaluate the reliability of the classifier. We present the concept of risk (or violation probability), which is a measure of the probability that a (new, unseen) sample is misclassified [14].

Definition 3: The probability (risk) of violating a constraint $g(\mathbf{X}_\iota, \mathbf{Y}_\iota)$, $\iota > N$, is denoted

$$R(\mathfrak{S}) := \mathbb{P}[\, g(\mathbf{X}_\iota, \mathbf{Y}_\iota) > 0 \mid (\mathbf{X}_\iota, \mathbf{Y}_\iota)\,], \tag{17}$$

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

We can provide a quantitative evaluation of the risk within the optimisation context of program (16): we mimic the discussion in [14] for binary SVM and adapt it to a multiclass environment.

Theorem 1 (Adapted From [14]): Given a confidence parameter $\beta \in [0, 1]$ and N samples, it holds that

$$\mathbb{P}^N[\underline{\epsilon}(s^*, N, \beta) \le R(\mathfrak{S}^*) \le \overline{\epsilon}(s^*, N, \beta)] > 1 - 3\beta, \tag{18}$$

where $\mathfrak{S}^*$ represents the parameters that minimise program (16) and s^* identifies the number of violated constraints, i.e., the number of samples that return $g(\mathbf{X}_i, \mathbf{Y}_i) > 0$. Further, the event of misclassification can be bounded by

$$\mathbb{P}^N[\mathbb{P}\ [\text{misclassification}] \le \overline{\epsilon}(s^*, N, \beta)] > 1 - 3\beta. \tag{19}$$

The bounds $\underline{\epsilon}$, $\overline{\epsilon}$ can be found solving a polynomial equation whose parameters are s^*, N, β, as outlined in [14].

The proof follows a similar result in [14] and is omitted for brevity.

Remark: It is worth highlighting that in program (16) the violation of a constraint does not imply misclassification: it reports that the evaluation of two hyperplanes differs less than ζ. As such, misclassification occurs more rarely than constraints violation, and Theorem 1 can solely provide an upper bound on the probability of misclassification, as stated by (19). Further, notice that the choice of the Veronese map relies on the prior knowledge about the linearity of (1) and the quadratic form of (2). The choice of kernel is aimed at minimising s^*, as $\overline{\epsilon}$ is proportional to it. Results with other kernels or formulations are matter of future work.

The hyperplanes defined by the classification algorithm outline the state-space partitions, and each partition corresponds to an abstract state of the ℓ-complete model. Given an initial state $\xi(0)$, we employ the classifier to map $\xi(0)$ to its corresponding abstract state, say x_0. We then evaluate the SAC and LAC that are reachable from x_0 in order to give upper and lower bounds for the AIST($\xi(0)$).

C. AIST Evaluation With Guarantees

The map $\mathcal{T}$ is equipped with probabilistic guarantees of correctness, which are delivered to the partitions of the concrete state space and thus to the abstract states of the Sℓ-CA. The domino transitions, on the other hand, are correct by design, albeit they might introduce spurious behaviours.

We do not know the number of ℓ-sequences that arises from (1)-(2). Assume $\ell = 2$, $\tau_i = \{1, 2\}$, and our dataset presents the 2-sequences $\{(1, 1), (1, 2), (2, 2)\}$ but not $(2, 1)$. We can gather a larger dataset until we observe the missing sequence; however, this might never happen as this particular 2-sequence may not be allowed by the triggering condition.

We synthesise a classifier based upon the dataset at hand: any new sample with an unseen label (not amongst the L labels) is evidently misclassified. We can thus rethink the misclassification of Theorem 1 as entailing both a mis-labeling – i.e., the algorithm assigns a wrong label to a sample – and a new labeling – i.e., a sample presents an unseen label. Hence we can write the bounds of Theorem 1 as

$$\mathbb{P}^N\big[\mathbb{P}\ [\text{mislabel}] + \mathbb{P}[\text{new label}] \leq \overline{\epsilon}(s^*, N, \beta)\big] > 1 - 3\beta. \tag{20}$$

In summary, after collecting N samples, we solely account for the labels present in the samples, set L as the number of *seen* labels, and proceed to solve optimisation problem (16).

Once the CM-SVM program (16) is solved, we assign a label to any new sample. Each label corresponds to an abstract state, which is used as initial state of the Sℓ-CA: we then compute the SAC and LAC (see (7), (8)) as lower and upper bounds for the corresponding AIST. The gap $\delta_{AIST} = LAC - SAC$ defines the precision of our abstraction: the true AIST dwells within the interval δ_{AIST}, therefore the tighter δ_{AIST} is, the more precise the information we gather. Recall that these results are tied to the abstraction: hence δ_{AIST} is correct with probability greater than $(1 - \overline{\epsilon})$, with confidence $(1 - 3\beta)$. If needed, we may increase ℓ to refine the abstraction. In this sense, ℓ can be seen as a tradeoff: increasing ℓ provides more information about the time-behaviour of the system and more labels. The SVM performance degrade with the increase of labels, hence we observe longer computational times and an increase in constraint violations s^* – which, in turn, degrades the probability bound $\overline{\epsilon}$. At the same time, more labels imply a larger abstract state space, returning a more precise abstraction (i.e., a smaller EAC).

TABLE I
PERFORMANCE OF THE CM-SVM ALGORITHM FOR THE 2D MODEL

ℓ	L	s^*	$\overline{\epsilon}$	Empir. Violat.	CM-SVM time [s]
1	3	99	0.020	1.14%	1.4
5	19	212	0.034	2.75%	8.7
10	34	325	0.048	3.90%	76.9

Further, in order to provide a global precision of the abstraction, we define the EAC (expected average cycle) as the average δ_{AIST} over all states; formally,

$$EAC := \frac{1}{|\mathcal{X}|} \sum_{\mathrm{x} \in \mathcal{X}} \delta_{AIST}(\mathrm{x}). \tag{21}$$

The EAC metric embodies the degree of uncertainty in the evaluation of the AIST. A smaller EAC ensures a more precise abstraction.

IV. EXPERIMENTAL EVALUATION

We show the effectiveness of our method considering a 2D, a 3D, and a 4D linear system, and comparing against a model-based technique, in order to validate our results and evaluate their precision and computational cost.

We collect $N = 10^4$ random samples $Z_i = (\mathbf{X}_i, \mathbf{Y}_i)$, evaluated from the 2D linear system in [3], [7], where

$$A = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & -5 \end{bmatrix}, \tag{22}$$

with triggering condition $|\xi(t) - \hat{\xi}(t)|^2 > \sigma^2 |\xi(t)|^2$, where $\sigma = 0.1$. We obtain a dataset with 3 possible ISTs, i.e., $\mathbf{Y}_i \in \{1, 2, 3\}$. We fix the hyper-parameters of the optimisation program (16) as $\beta = 10^{-6}$, $\rho = 10^3$.

The training data is used to solve the optimisation problem (16) and obtain the mapping from the concrete states $\xi(\cdot)$ to the abstract states x. We collect the number of violated constraints s^* and compute the corresponding $\overline{\epsilon}$ bound according to (19). Table I reports the number of labels L, the violated constraints s^*, its corresponding bound $\overline{\epsilon}$ and the empirical percentage of violated constraints (computed on additional 10^4 samples) and finally the computational time of the CM-SVM algorithm, for different values of ℓ. As expected, the empirical constraint violation percentage is smaller than $\overline{\epsilon}$ for all experiments. The number of samples impacts the results: as proved in [14], $\overline{\epsilon}$ is proportional to $2/N$, s^* and $\log(\beta)$. Tighter bounds can obtained by increasing the number of samples, likely deteriorating the time performance.

We then construct the abstraction: considering several values of ℓ, we report the number of states $|\mathcal{X}|$, transitions $|\mathcal{E}|$, the EAC in Table II. The increase of ℓ, together with an obvious increase in the number of states and transitions, provides a smaller EAC, in view of the decrease of the abstraction's non-determinism (the number of states approaches the number of transitions). We challenge our data-driven methodology against a model-based procedure developed in [3], [7], [8]. Comparing the two abstractions, our procedure returns an

TABLE II
DATA-DRIVEN AND MODEL-BASED ABSTRACTIONS FOR THE 2D (TOP), 3D (MIDDLE), 4D (BOTTOM) MODELS. TIME IS IN SECONDS

ℓ	Data-driven					Model-based			
	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{E}\rvert$	$\bar{\epsilon}$	EAC	Time	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{E}\rvert$	EAC	Time
1	3	9	0.010	2.0	1	3	9	2.0	6
5	18	24	0.016	2.0	5	7	17	2.0	4
10	33	38	0.020	1.0	60	34	39	0.97	3
1	3	9	0.011	2.0	2	3	9	2.0	9
2	7	17	0.013	2.0	6	7	17	2.0	7
3	13	26	0.014	2.0	16	15	33	2.0	18
4	20	34	0.015	1.5	22	26	46	2.0	59
1	10	100	0.061	9.0	9	10	100	9.0	6
2	43	218	0.002	3.5	26	-	-	-	TO
3	112	291	0.002	2.35	113	-	-	-	TO
4	223	463	0.002	2.27	329	-	-	-	TO

almost equivalent transition system, smaller by few states (and transitions): being sample-based, our abstraction may lack the states belonging to a small, arguably negligible, portion of the state-space. Nevertheless, the EAC remains very close to the model-based one.

We finally challenge our procedure with a 3D and a 4D models as proposed in [4], [7]. For this case study we generate $2 \cdot 10^4$ and $5 \cdot 10^4$ samples, respectively. We apply our procedure for $\ell = [1, 4]$, and report the results in Table II (middle and bottom). We notice that the computational time of our procedure remains reasonable for every test, despite the large number of labels. The model-based procedure, on the other hand, exploits the knowledge of the system model to eliminate the transitions and states that are not permitted by the triggering condition. This inspection increases the procedure time, until it reaches the time out of 10 minutes.

As noted in [16], the cone related to a particular ℓ-sequence may correspond to a zero-measure set (i.e., the cone reduces to a line). The model-based procedure includes the zero-measure cones in its computation of the SAC and LAC, whereas the data-driven procedure has zero probability to sample and account for these. In this sense, the data-driven EAC computation is robust to this degenerate case. Our framework can be applied to a noisy model (1): this is matter of future research. Noisy data likely increases the computational time and the number of violated constraints s^*, which in turn degrade the probability bound $\bar{\epsilon}$.

V. CONCLUSION AND FUTURE WORK

We have presented a method to estimate the sampling performance of an unknown PETC system, namely its average inter-sample time, by means of a data-driven abstraction. We extend the scenario approach to multiclass SVM algorithms and build an ℓ-complete abstraction to return bounds on the AIST. We challenge our procedure against model-based state-of-the-art tools: the data-driven approach is computationally faster for high dimensional systems whilst providing tight probabilistic guarantees. Future work includes the application of this methodology to noisy, nonlinear systems. We also aim at extending the scenario approach to neural classifiers to overcome the limitations of SVM.

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*, vol. 2, 2002, 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] G. D. A. Gleizer and M. Mazo, Jr., "Towards traffic bisimulation of linear periodic event-triggered controllers," *IEEE Contr. Syst. Lett.*, vol. 5, pp. 25–30, 2021.

[4] G. D. A. Gleizer and M. Mazo, Jr., "Scalable traffic models for scheduling of linear periodic event-triggered controllers," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2726–2732, 2020.

[5] A. S. Kolarijani and M. Mazo, Jr., "Formal traffic characterization of LTI event-triggered control systems," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 1, pp. 274–283, Mar. 2018.

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

[7] 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.

[8] 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.

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

[10] T. S. Badings, A. Abate, N. Jansen, D. Parker, H. A. Poonawala, and M. Stoelinga, "Sampling-based robust control of autonomous systems with non-Gaussian noise," 2021, *arXiv:2110.12662*.

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

[12] 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.

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

[14] 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 (CDC)*, 2020, pp. 2463–2468.

[15] 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.

[16] G. de Albuquerque Gleizer and M. Mazo, Jr., "Chaos and order in event-triggered control," 2022, *arXiv:2201.04462*.

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

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

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

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

[21] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[22] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Mach. Learn. Res.*, vol. 2, pp. 265–292, Mar. 2002.