Data-Driven Models of Monotone Systems

Anas Makdesi, Antoine Girard, Laurent Fribourg

Data-Driven Models of Monotone Systems

Anas Makdesi, Antoine Girard, *Senior Member, IEEE*, and Laurent Fribourg

***Abstract*— In this paper, we consider the problem of computing from data guaranteed set-valued over-approximations of unknown monotone functions with additive disturbances. We provide a characterization of a simulating map that provably contains all monotone functions that are consistent with the data. This map is also minimal in the sense that any set-valued map containing all consistent monotone functions would also include the map we are proposing. We show that this minimal simulating map is interval-valued and admits a simple construction on a finite partition induced by the data. As the complexity of the partition increases with the amount of data, we also consider the problem of computing minimal interval-valued simulating maps defined on partitions that are fixed a priori. We present an efficient algorithm for their computation. We then use those data-driven over-approximations to build models for partially unknown systems where the unknown part is monotone. The resulting models are used to construct finite-state symbolic abstractions, paving the way for discrete controller synthesis methods to be applied. We extend our approach to handle systems with bounded derivatives and introduce an algorithm to calculate the bounds on those derivatives and on the disturbances from the data. We present several numerical experiments to test the performance of the introduced method and show that the data-driven abstractions are suitable for controller synthesis purposes.**

***Index Terms*— Monotone maps, Monotone systems, Data-driven models, Data-driven abstraction, Symbolic control.**

I. Introduction

The recent advancements in data acquisition tools, coupled with the ability to efficiently deal with the ever-increasing amount of data harnessed by those tools, gave rise to many new data-driven approaches and techniques in control theory. Those approaches have the benefit of being able to tackle unknown or hard-to-model systems, which justifies the increasing interest in them. While some methods rely on finding the controller directly from the data [12, and references therein], others depend on finding data-driven models, which then can be used to find the controllers. A survey of both methods can be found in [2]. Learning the dynamics to find systems' models leverages the ability to use well-established, well-studied approaches to find controllers enforcing the desired behaviors. Moreover, data-driven models can sometimes come with formal guarantees regarding the approximation of the real system. Such guaranteed models can be categorized as stochastic or robust [11, and references therein]. The present work follows the latter approaches, where the aim is to find bounds on the function representing the system (non-parametric approach) or bounds on some unknown parameters of the function representing the system (parametric approach).

The set membership approach offers an example of how we can find those robust models. In [27], the set membership approach is used for the identification of nonlinear systems. Set membership approaches are used in [4] to synthesize nonlinear model predictive controllers. In contrast to [4], the introduced approach does not assume anything about the stability of the unknown systems. Also, the number of points that can be handled is greater than the number addressed in the set membership approach. Another example of robust models can be found in [17], where interval observers for partially unknown mixed-monotone nonlinear systems are found, assuming the unknown part is Lipschitz continuous. Lipschitz dynamical systems with known bounds on the Lipschitz constants are also studied in [30] to reach piecewise affine set-valued models. In contrast to [17] and [30], we do not consider the case of Lipschitz unknown functions. Instead, we start by studying monotone functions. Relying on the monotonicity assumption, we developed an efficient method to handle a large number of data points. Monotone functions can be found in many applications, such as adaptive cruise control [13] or power networks [36]. In [25], models that enforce a notion of monotonicity are learned following the argument that, for many machine learning problems, some inputs relate to the output monotonically, such as house pricing. Data-driven approaches are used in [15] for monotone systems reduction.

In this work, given a set of data generated from an unknown monotone function with an additive bounded disturbance, we find an over-approximating set-valued map, which we call a simulating map, guaranteed to include any monotone function capable of generating the data. Moreover, we look for the tightest map with this guarantee. The resulting model (the minimal simulating map) can be naturally defined as an interval-valued map using a partition of the input space, which only depends on the sampled data, and thus it is non-parametric. We also consider a parametric case where a partition of the input space is given a priori. We find the tightest interval-valued over-approximation on this partition. Although fixing the partition introduces more conservatism to the resulting models, those models can be calculated and handled more efficiently.

This project has received funding from the H2020-EU.1.1. research and innovation programme – ERC-2016-COG under grant agreement No 725144.

A. Makdesi and A. Girard are with Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France (e-mail: Anas.Makdesi@l2s.centralesupelec.fr, Antoine.Girard@l2s.centralesupelec.fr).

L. Fribourg is with Université Paris-Saclay, CNRS, ENS Paris-Saclay, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France (e-mail: fribourg@lsv.ens-cachan.fr).

Then, we use our over-approximation method to learn models of discrete-time dynamical systems with an unknown monotone part. Those models are then used to find finite-state symbolic models of the systems. Symbolic control is a computational approach to controller synthesis where discrete abstractions (symbolic models) of continuous-state dynamical systems are calculated. Then, discrete controller synthesis techniques [1], [33] are used to find correct-by-design controllers; those controllers are capable of ensuring complex specifications (safety and reachability [7], behaviors described by automata [28], or temporal logic formulas [1]). In the case of an unknown monotone system, we show that working with the finite-state symbolic models does not add any conservatism compared to working with the tightest data-driven over-approximation models.

Although symbolic control is usually presented as a model-driven technique, some data-driven approaches have emerged recently. For instance, approaches that only require the ability to sample the system dynamics on a given grid of states and inputs are introduced in [9], [24], or [35]. In [6], data-driven abstractions within the PAC (probably approximately correct) statistical framework are found. PAC guarantees are also used in [5] to build data-driven abstraction through the use of the scenario approach. The scenario approach is also used in [18] and [16] to offer probabilistic guarantees on the data-driven built abstractions. The main difference between all the previously mentioned work and the approach introduced here is that, in contrast to our approach, they all give probabilistic guarantees. Data-driven control design with regular language specifications is studied in [29] for plants described as abstract systems. In comparison to [29], our approach can handle bounded disturbances and is able to satisfy the specifications robustly, not to a predefined desired accuracy. Unknown parts of nonlinear systems are modeled using Gaussian processes; then, those models are used in building symbolic abstractions in [10].

Some results in this paper appeared in preliminary form in the conference papers [21], [20], where we focused on the case of pure monotone systems. The present paper brings a globally improved presentation of our approach with a redesigned concept of minimal simulating maps and complete proofs of theoretical results (whereas only sketches of proofs, in the undisturbed case, were provided in our previous works). In addition, we introduce a new efficient algorithm to compute our set-valued over-approximations. We also generalize the approach to the case of general systems with an unknown monotone part. Using ideas inspired by those introduced to study mixed-monotone systems in [34], we extend our approach to study systems with bounded derivatives. Finally, we use the scenario approach, developed for robust control design in [3], to estimate upper and lower bounds on the partial derivatives of the unknown function generating the data. Finally, we validate our results with more extensive numerical experiments.

This paper is organized as follows. In Section II, we introduce the problem of over-approximating monotone maps, give the necessary definitions, and then compute the optimal solution, and optimal solution given a fixed partition a priori. In the case of fixed partition, we provide an efficient algorithm to calculate the over-approximation. In Section III, we use the introduced over-approximation to build data-driven models for systems with unknown monotone parts. Then, we use those data-driven models to find a finite-state symbolic representation of the system. We also show how we can study systems with bounded derivatives in this section. In Section IV, we provide a way to find the upper and lower bounds on the partial derivatives of a function using data generated by this function. Finally, Section V is dedicated to presenting some numerical examples.

Notations: $\overline{\mathbb{R}} = [-\infty, +\infty]$ is the set of *extended real numbers*. We use bold lowercase letters to represent vectors, e.g. $\mathbf{z} \in \overline{\mathbb{R}}^n$; subscripts are used to differentiate between multiple vectors $\mathbf{z}_i$, whereas normal lowercase letters with superscripts z^i are used to denote the i^{th} component of a vector $\mathbf{z}$. Given two vectors $\mathbf{z}_1, \mathbf{z}_2 \in \overline{\mathbb{R}}^n$, we define the partial order $\preceq$ on $\overline{\mathbb{R}}^n$ to be $\mathbf{z}_1 \preceq \mathbf{z}_2$ if and only if $z_1^i \leq z_2^i$ for all $i = 1, \ldots, n$. We use indifferently $\mathbf{z}_1 \preceq \mathbf{z}_2$ and $\mathbf{z}_2 \succeq \mathbf{z}_1$. $[\mathbf{z}_1, \mathbf{z}_2] = \{\mathbf{z} \in \overline{\mathbb{R}}^n \,|\, \mathbf{z}_1 \preceq \mathbf{z} \preceq \mathbf{z}_2\}$ defines a closed *interval* of $\overline{\mathbb{R}}^n$. We define $\max(\mathbf{z}_1, \mathbf{z}_2)$, or $\min(\mathbf{z}_1, \mathbf{z}_2)$, to be the vector $\mathbf{z}$ whose components are $z^i = \max(z_1^i, z_2^i)$, or $z^i = \min(z_1^i, z_2^i)$ respectively. Given a set $Z \subseteq \overline{\mathbb{R}}^n$, $\operatorname{int} Z$ and $\operatorname{cl} Z$ denote the *interior* and the *closure* of the set Z, $\inf Z$ and $\sup Z$ denote the *infimum* and the *supremum* of Z, i.e. the greatest lower and least upper bounds of Z relative to partial order $\preceq$ on $\overline{\mathbb{R}}^n$. Given a collection of sets $Z_k \subseteq \overline{\mathbb{R}}^n$, for k in some index set, we will use the following convention $\bigcap_{k \in \emptyset} Z_k = \overline{\mathbb{R}}^n$. For $\mathbf{z} \in \overline{\mathbb{R}}^n$ and $Z \subseteq \overline{\mathbb{R}}^n$, we define $\mathbf{z} + Z = \{\mathbf{z} + \mathbf{z}' \,|\, \mathbf{z}' \in Z\}$. A relation $R \subseteq Z \times Y$ is identified with the set-valued map $R : Z \rightrightarrows Y$ where $R(\mathbf{z}) = \{\mathbf{y} \in Y \,|\, (\mathbf{z}, \mathbf{y}) \in R\}$. We define the converse relation $R^{-1} = \{(\mathbf{y}, \mathbf{z}) \in Y \times Z \,|\, (\mathbf{z}, \mathbf{y}) \in R\}$. A single-valued map $f : Z \to Y$ can be identified to the deterministic set-valued map $F : Z \rightrightarrows Y$ where $F(\mathbf{z}) = \{f(\mathbf{z})\}$ for all $\mathbf{z} \in Z$. In this paper, to avoid confusion, we will refer to set-valued maps as *maps* and to single-valued maps as *functions*. A map $F : Z \rightrightarrows Y$ where $Y \subseteq \overline{\mathbb{R}}^m$ is called an *interval-valued map* if for all $\mathbf{z} \in Z$, there exists $\mathbf{y}_1, \mathbf{y}_2 \in Y$ such that $F(\mathbf{z}) = [\mathbf{y}_1, \mathbf{y}_2]$. An *over-approximation* of a map $F : Z \rightrightarrows Y$ is a map $\hat{F} : Z \rightrightarrows Y$ such that $F(\mathbf{z}) \subseteq \hat{F}(\mathbf{z})$, for all $\mathbf{z} \in Z$. A property is true for *almost all* $\mathbf{z} \in Z$ if it holds for all $\mathbf{z} \in Z \setminus Z_0$ where $Z_0 \subseteq Z$ is of measure zero. Then given two maps $F_1, F_2 : Z \rightrightarrows Y$, we say that $F_1 = F_2$ *almost everywhere* (a.e.) if $F_1(\mathbf{z}) = F_2(\mathbf{z})$ for almost all $\mathbf{z} \in Z$.

II. Learning Monotone Maps from Data

In this section, we consider the problem of learning monotone maps from data. For a class of monotone maps, we provide an approach to compute over-approximations of the maps that are as tight as possible, given the available data.

As we are dealing with set-valued maps, it is adequate to properly define the notion of monotonicity for such maps.

Definition 1: The map $F : Z \rightrightarrows Y$, with $Z \subseteq \overline{\mathbb{R}}^n$ and $Y \subseteq \overline{\mathbb{R}}^m$, is *monotone* if for all $\mathbf{z}, \mathbf{z}' \in Z$ with $\mathbf{z} \preceq \mathbf{z}'$,

$$\forall \mathbf{y} \in F(\mathbf{z}),\ \exists \mathbf{y}' \in F(\mathbf{z}'),\ \mathbf{y} \preceq \mathbf{y}',\ \text{and}$$
$$\forall \mathbf{y}' \in F(\mathbf{z}'),\ \exists \mathbf{y} \in F(\mathbf{z}),\ \mathbf{y} \preceq \mathbf{y}'.$$

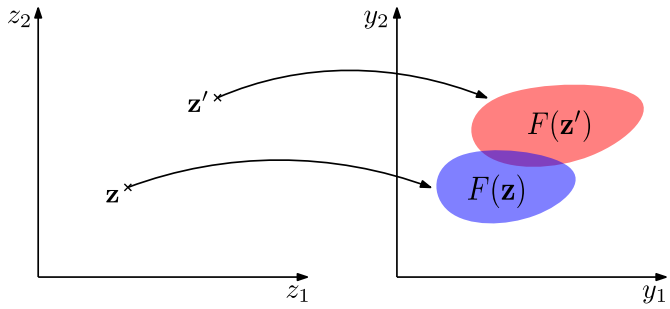

Fig. 1. Example of a set-valued monotone map $\boldsymbol{F : Z \rightrightarrows Y}$.

An illustration of the notion of monotone maps is shown in Figure 1. Let us remark that in the case of a function $f : Z \to Y$, Definition 1 coincides with the usual definition of monotone functions, that is

$$\forall \mathbf{z}, \mathbf{z}' \in Z,\ \mathbf{z} \preceq \mathbf{z}' \implies f(\mathbf{z}) \preceq f(\mathbf{z}').$$

This paper considers a class of maps given by monotone functions with additive bounded disturbances. Formally, let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ where $Z \subseteq \overline{\mathbb{R}}^n$ and such that

$$\forall \mathbf{z} \in Z,\ F(\mathbf{z}) = f(\mathbf{z}) + W, \tag{1}$$

where $f : Z \to \overline{\mathbb{R}}^m$ is a monotone function and $W = [\underline{\mathbf{w}}, \overline{\mathbf{w}}]$, $W \subseteq \mathbb{R}^m$ is a bounded interval of disturbances. The following property of F is a straightforward consequence of (1) and of Definition 1 and is therefore stated without proof.

Claim 1: The map F given by (1), where f is a monotone function and $W \subseteq \mathbb{R}^m$, is a monotone map.

We can now describe the problem under consideration in this section:

- Let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ of the form (1) where the monotone function $f : Z \to \overline{\mathbb{R}}^m$ is **unknown** and the disturbances lower and upper bounds $\underline{\mathbf{w}}, \overline{\mathbf{w}} \in \mathbb{R}^m$ are **known**.
- Let us consider a set of data $\mathcal{D} \subseteq Z \times \overline{\mathbb{R}}^m$ generated by the map F:

$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), k \in \mathbb{K}\}, \tag{2}$$

  where $\mathbb{K}$ is a finite set of indices.

Given the data $\mathcal{D}$, the bounds $\underline{\mathbf{w}}$, $\overline{\mathbf{w}}$, and under the sole assumption that f is monotone, we aim at computing an over-approximation of the map F that is as "tight" as possible. A precise notion of tightness will be formally defined in subsection II-B.

Remark 1: In practical applications, we are mostly interested in maps $F : Z \rightrightarrows \mathbb{R}^m$ with $Z \subseteq \mathbb{R}^n$. Note that any such map can be seen as a map from a subset of $\overline{\mathbb{R}}^n$ to $\overline{\mathbb{R}}^m$ and can thus be cast in our framework. Moreover, working with extended real numbers will facilitate the mathematical analysis of our approach.

A. Consistent and simulating maps

Let us define several notions that will help us formalize our problem. A map will be said to be consistent if it is of the form (1) and is capable of generating the data $\mathcal{D}$.

Definition 2: A map $\tilde{F} : Z \rightrightarrows \overline{\mathbb{R}}^m$ is *consistent* with the data $\mathcal{D}$ if the following hold:

1) There exists a monotone function $\tilde{f} : Z \to \overline{\mathbb{R}}^m$ such that, for all $\mathbf{z} \in Z$, $\tilde{F}(\mathbf{z}) = \tilde{f}(\mathbf{z}) + W$;
2) For all $(\mathbf{z}, \mathbf{y}) \in \mathcal{D}$, $\mathbf{y} \in \tilde{F}(\mathbf{z})$.

We denote the set of maps that are consistent with the data $\mathcal{D}$ by $\mathbb{C}_{\mathcal{D}}$.

Obviously, there is at least one map that is consistent with $\mathcal{D}$, which is the map F that generated $\mathcal{D}$. In general, there could be more. Then, a map over-approximating all the consistent maps is called a simulating map.

Definition 3: A map $S : Z \rightrightarrows \overline{\mathbb{R}}^m$ is a *simulating map* of the data $\mathcal{D}$ if for all $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$, for all $\mathbf{z} \in Z$, $\tilde{F}(\mathbf{z}) \subseteq S(\mathbf{z})$. We denote the set of all simulating maps of $\mathcal{D}$ by $\mathbb{S}_{\mathcal{D}}$.

A trivial and useless example of a simulating map is the map given for all $\mathbf{z} \in Z$ by $S(\mathbf{z}) = \overline{\mathbb{R}}^m$. Out of the maybe infinite number of simulating maps, we are looking for the tightest ones, which we call minimal simulating maps. Those maps are the least conservative estimate of the actual map F, given the available data $\mathcal{D}$. Formally, they are defined as follows:

Definition 4: A map $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$ is a *minimal simulating map* of the data $\mathcal{D}$ if the following hold:

1) $S_m \in \mathbb{S}_{\mathcal{D}}$ (Simulation);
2) For almost all $\mathbf{z} \in Z$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m(\mathbf{z}) \subseteq S(\mathbf{z})$ (Minimality).

Remark 2: It is important to emphasize that while minimality holds only almost everywhere, it is required that the simulation property holds for all $\mathbf{z} \in Z$. Hence, a minimal simulating map is always an over-approximation of the unknown map F that generated the data $\mathcal{D}$. The main reason for requiring minimality only almost everywhere is that it will make it possible to give a simple construction of the minimal simulating map as shown in subsection II-B.

In subsection II-B, we will provide a constructive proof of the existence of minimal simulating maps. The following proposition describes the relation between minimal simulating maps.

Proposition 1: Given two maps $S_m, S'_m : Z \rightrightarrows \overline{\mathbb{R}}^m$, the following properties hold:

- If S_m and S'_m are minimal simulating maps of $\mathcal{D}$, then $S_m = S'_m$ a.e.;
- If S_m is a minimal simulating map of $\mathcal{D}$, $S'_m \in \mathbb{S}_{\mathcal{D}}$ and $S_m = S'_m$ a.e., then S'_m is a minimal simulating map of $\mathcal{D}$.

Proof: We prove the first part of the proposition. According to Definition 4, there exists a set $Z_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0$, $S_m(\mathbf{z}) \subseteq S'_m(\mathbf{z})$, and there exists a set $Z'_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z'_0$, $S'_m(\mathbf{z}) \subseteq S_m(\mathbf{z})$. Hence, for all $\mathbf{z} \in Z \setminus (Z_0 \cup Z'_0)$, $S'_m(\mathbf{z}) = S_m(\mathbf{z})$. The set $Z_0 \cup Z'_0$ being of measure zero, it follows that $S_m = S'_m$ a.e..

Now, let us show the second part of the proposition. For S'_m to be a minimal simulating map, S'_m should meet the two requirements of Definition 4. The first one is satisfied by

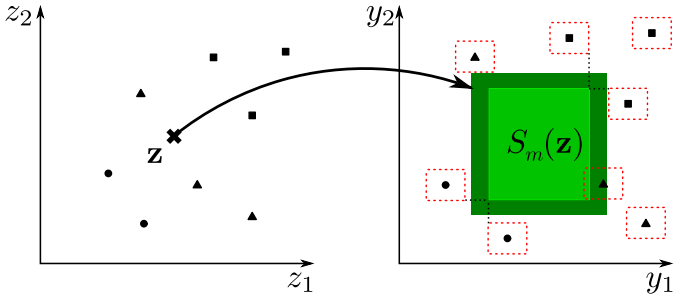

Fig. 2. Construction of a minimal simulating map. Left: data points in the input space $\boldsymbol{z_k} \in \boldsymbol{Z}$, $\boldsymbol{k} \in \mathbb{K}$, points larger than ($\boldsymbol{k} \in \mathbb{K}^+(\mathbf{z})$) / lesser than ($\boldsymbol{k} \in \mathbb{K}^-(\mathbf{z})$) / incomparable to a given $\mathbf{z} \in \boldsymbol{Z}$ are represented with squares / circles / triangles, respectively. Right: data points in the output space $\boldsymbol{y_k} \in \overline{\mathbb{R}}^{\boldsymbol{m}}$, $\boldsymbol{k} \in \mathbb{K}$ are surrounded by red intervals representing the bounded disturbance, and the green interval represents the minimal simulating map $\boldsymbol{S_m(z)} \subseteq \overline{\mathbb{R}}^{\boldsymbol{m}}$ given by (3). The light green area represents the approximation of the function $\boldsymbol{f}$ as mentioned in Remark 3.

assumption. The second one can be shown as follows. There exists a set $Z_0 \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m(\mathbf{z}) \subseteq S(\mathbf{z})$. Also, there exists a set $Z_0' \subseteq Z$ of measure zero such that, for all $\mathbf{z} \in Z \setminus Z_0'$, $S_m'(\mathbf{z}) = S_m(\mathbf{z})$. Hence, for all $\mathbf{z} \in Z \setminus (Z_0 \cup Z_0')$, for all $S \in \mathbb{S}_{\mathcal{D}}$, $S_m'(\mathbf{z}) \subseteq S(\mathbf{z})$. Because $Z_0 \cup Z_0'$ is of measure zero, the second requirement of Definition 4 is satisfied, and S_m' is a minimal simulating map. ■

Proposition 1 essentially states that minimal simulating maps are uniquely determined up to a set of measure zero. We can now formally state the problem under consideration in this section:

Problem 1: Given the data $\mathcal{D}$ and the disturbance bounds $\underline{\mathbf{w}}$, $\overline{\mathbf{w}} \in \mathbb{R}^m$, compute $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$, a minimal simulating map of $\mathcal{D}$.

B. Computation of minimal simulating maps

In the following, we describe a solution to Problem 1. We first establish a characterization of a minimal simulating map and prove some of its properties. Then, a practical approach is provided for computing a minimal simulating map.

1) Characterization and properties: The following theorem provides an effective characterization of a minimal simulating map:

Theorem 1: Let $S_m : Z \rightrightarrows \overline{\mathbb{R}}^m$ be the map given for all $\mathbf{z} \in Z$ by:

$$S_m(\mathbf{z}) = \left(\bigcap_{k \in \mathbb{K}^-(\mathbf{z})} \left\{ \mathbf{y} \in \overline{\mathbb{R}}^m \mid \mathbf{y}_k + \underline{\mathbf{w}} - \overline{\mathbf{w}} \preceq \mathbf{y} \right\} \right) \cap \left(\bigcap_{k \in \mathbb{K}^+(\mathbf{z})} \left\{ \mathbf{y} \in \overline{\mathbb{R}}^m \mid \mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{w}} - \underline{\mathbf{w}} \right\} \right), \tag{3}$$

where

$$\mathbb{K}^-(\mathbf{z}) = \{k \in \mathbb{K} \mid \mathbf{z}_k \preceq \mathbf{z}\}, \ \mathbb{K}^+(\mathbf{z}) = \{k \in \mathbb{K} \mid \mathbf{z} \preceq \mathbf{z}_k\}. \tag{4}$$

Then, S_m is a minimal simulating map of $\mathcal{D}$.

Figure 2 shows the construction of the map S_m. Let us remark that Theorem 1 also provides a constructive proof of the existence of minimal simulating maps. It follows from (3) that the minimal simulating map S_m is interval-valued. Indeed, for all $\mathbf{z} \in Z$, $S_m(\mathbf{z}) = [\underline{S}_m(\mathbf{z}), \overline{S}_m(\mathbf{z})]$ with upper and lower bounds $\overline{S}_m(\mathbf{z})$, $\underline{S}_m(\mathbf{z})$ given by

$$\overline{S}_m(\mathbf{z}) = \inf \left\{ \mathbf{y}_k + \overline{\mathbf{w}} - \underline{\mathbf{w}} \mid k \in \mathbb{K}^+(\mathbf{z}) \right\}, \tag{5}$$

$$\underline{S}_m(\mathbf{z}) = \sup \left\{ \mathbf{y}_k + \underline{\mathbf{w}} - \overline{\mathbf{w}} \mid k \in \mathbb{K}^-(\mathbf{z}) \right\}. \tag{6}$$

We now prove Theorem 1.

Proof: Let us first show that $S_m \in \mathbb{S}_{\mathcal{D}}$. Let $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$, then there exists a monotone function $\tilde{f} : Z \to \overline{\mathbb{R}}^m$ such that, $\tilde{F}(\mathbf{z}) = \tilde{f}(\mathbf{z}) + W$. Let $\mathbf{z} \in Z$, $\mathbf{y} \in \tilde{F}(\mathbf{z})$, then, $\mathbf{y} \preceq \tilde{f}(\mathbf{z}) + \overline{\mathbf{w}}$. From the monotonicity of $\tilde{f}$, we have $\tilde{f}(\mathbf{z}) \preceq \tilde{f}(\mathbf{z}_k)$ for all $k \in \mathbb{K}^+(\mathbf{z})$. Therefore,

$$\mathbf{y} \preceq \tilde{f}(\mathbf{z}_k) + \overline{\mathbf{w}}, \ \forall k \in \mathbb{K}^+(\mathbf{z}).$$

Moreover, $\tilde{F} \in \mathbb{C}_{\mathcal{D}}$ implies $\mathbf{y}_k \in \tilde{F}(\mathbf{z}_k)$, for all $k \in \mathbb{K}$. Therefore, we have

$$\tilde{f}(\mathbf{z}_k) + \underline{\mathbf{w}} \preceq \mathbf{y}_k, \ \forall k \in \mathbb{K}.$$

Hence, we get from the inequalities above that

$$\mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{w}} - \underline{\mathbf{w}}, \ \forall k \in \mathbb{K}^+(\mathbf{z}).$$

Then, from (5), $\mathbf{y} \preceq \overline{S}_m(\mathbf{z})$. Similarly, it can be shown that $\mathbf{y} \succeq \underline{S}_m(\mathbf{z})$. Therefore, we have $\mathbf{y} \in S_m(\mathbf{z})$; hence, S_m is a simulating map of $\mathcal{D}$.

Now, we prove minimality. For any arbitrary point included in S_m we will build a consistent map containing this point. Consider the set Z_0 of measure zero, defined as

$$Z_0 = \{\mathbf{z} \in Z \mid \exists i \in \{1, \ldots, n\}, k \in \mathbb{K}, z^i = z_k^i\}. \tag{7}$$

This set consists of the union of finitely many hyperplanes. Each one of those hyperplanes is defined by one of the components of a data point. According to Definition 4, minimality has to hold almost everywhere. Then, let us consider $\mathbf{z}^\star \in Z \setminus Z_0$ and $\mathbf{y}^\star \in S_m(\mathbf{z}^\star)$. Let us define a partition of Z in 3 regions defined by

$$Z_1 = Z^+(\mathbf{z}^\star) \setminus Z^-(\mathbf{z}^\star), \ Z_2 = Z^+(\mathbf{z}^\star) \cap Z^-(\mathbf{z}^\star), \ Z_3 = Z \setminus (Z_1 \cup Z_2),$$

where

$$Z^-(\mathbf{z}^\star) = \bigcap_{i=1}^n \left\{ \mathbf{z} \in Z \mid z^i > \sup\{z_k^i \mid z_k^i \leq z^{\star i}, k \in \mathbb{K}\} \right\},$$

$$Z^+(\mathbf{z}^\star) = \bigcap_{i=1}^n \left\{ \mathbf{z} \in Z \mid z^i < \inf\{z_k^i \mid z_k^i \geq z^{\star i}, k \in \mathbb{K}\} \right\}.$$

A representation of the regions can be found in Figure 3. Let us remark that by construction $\mathbf{z}_k \in Z_1 \cup Z_3$ for all $k \in \mathbb{K}$ and that $\mathbf{z}^\star \in Z_2$ because $\mathbf{z}^\star \in Z \setminus Z_0$.

Then, let us consider the map $F^\star : Z \rightrightarrows \overline{\mathbb{R}}^m$ given for all $\mathbf{z} \in Z$ by $F^\star(\mathbf{z}) = f^\star(\mathbf{z}) + W$ where $f^\star : Z \to \overline{\mathbb{R}}^m$ is defined as follows

$$f^\star(\mathbf{z}) = \begin{cases} \underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}} & \text{if } \mathbf{z} \in Z_1, \\ \max(\min(\mathbf{y}^\star, \overline{S}_m(\mathbf{z}^\star) - \overline{\mathbf{w}}), \underline{S}_m(\mathbf{z}^\star) - \underline{\mathbf{w}}) & \text{if } \mathbf{z} \in Z_2, \\ \overline{S}_m(\mathbf{z}) - \overline{\mathbf{w}} & \text{if } \mathbf{z} \in Z_3. \end{cases} \tag{8}$$

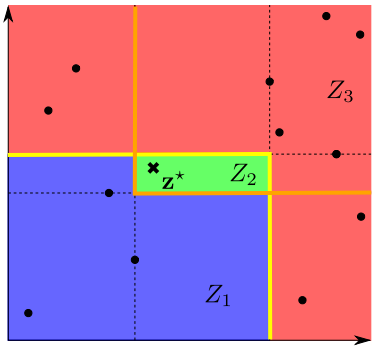

Fig. 3. Partition of $\boldsymbol{Z}$ used to define the function $\boldsymbol{f^\star}$ in (8). The area bounded from above by the yellow line represents the set $\boldsymbol{Z^+(\mathbf{z}^\star)}$. The area bounded from below by the orange line represents the set $\boldsymbol{Z^-(\mathbf{z}^\star)}$. The area $\boldsymbol{Z_1}$ in blue contains all the data points smaller than $\mathbf{z}^\star$. The area $\boldsymbol{Z_3}$ in red contains all the data points larger than $\mathbf{z}^\star$ or incomparable with $\mathbf{z}^\star$. The area $\boldsymbol{Z_2}$ is colored in green.

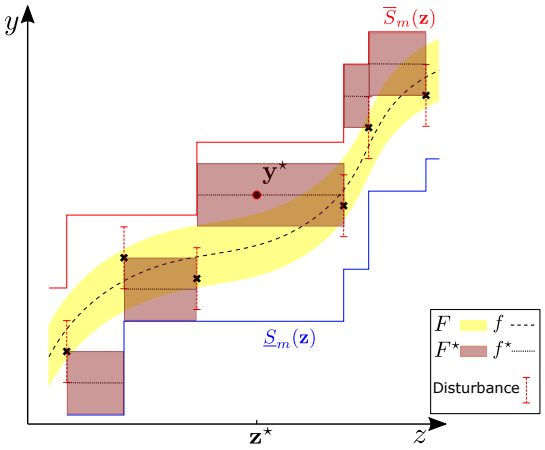

Fig. 4. Construction of the map $\boldsymbol{F^\star}$ according to (8).

In Figure 4, a representation, in the one-dimensional case, of the maps $F, F^\star$, and S_m can be found. Since $\mathbf{z}^\star \in Z_2$ and since $\mathbf{y}^\star \in S_m(\mathbf{z}^\star) = [\underline{S}_m(\mathbf{z}^\star), \overline{S}_m(\mathbf{z}^\star)]$, it follows from (8) that $\mathbf{y}^\star \in F^\star(\mathbf{z}^\star)$. We will now prove that $F^\star \in \mathbb{C}_\mathcal{D}$.

We first prove that function $f^\star$ is monotone. Let $\mathbf{z}, \mathbf{z}' \in Z$ such that $\mathbf{z} \preceq \mathbf{z}'$. There are several possibilities for the position of $\mathbf{z}$ and $\mathbf{z}'$: $i)$ $\mathbf{z}, \mathbf{z}' \in Z_1$, $ii)$ $\mathbf{z}, \mathbf{z}' \in Z_2$, $iii)$ $\mathbf{z}, \mathbf{z}' \in Z_3$, $iv)$ $\mathbf{z} \in Z_1, \mathbf{z}' \in Z_2$, $v)$ $\mathbf{z} \in Z_2, \mathbf{z}' \in Z_3$, $vi)$ $\mathbf{z} \in Z_1, \mathbf{z}' \in Z_3$. We want to verify that $f^\star(\mathbf{z}) \preceq f^\star(\mathbf{z}')$ in all those cases.

- Case (i): Since $\mathbf{z} \preceq \mathbf{z}'$, we have $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$, which implies from (6), $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}')$. Then, from (8), $f^\star(\mathbf{z}) \preceq f^\star(\mathbf{z}')$.
- Case (ii): From (8), we have $f^\star(\mathbf{z}) = f^\star(\mathbf{z}')$.
- Case (iii): Similar to case (i).
- Case (iv): It follows from the definition of Z_2 that $\mathbb{K}^-(\mathbf{z}') = \mathbb{K}^-(\mathbf{z}^\star)$. Then, since $\mathbf{z} \preceq \mathbf{z}'$, we have $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$ and thus $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}^\star)$. Hence, from (6), we get $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}^\star)$. Since $\mathbf{z} \in Z_1$ and $\mathbf{z}' \in Z_2$ we get from (8),
$$f^\star(\mathbf{z}) = \underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}} \preceq \underline{S}_m(\mathbf{z}^\star) - \underline{\mathbf{w}} \preceq f^\star(\mathbf{z}').$$
- Case (v): Similar to case (iv).
- Case (vi): Since $S_m \in \mathbb{S}_\mathcal{D}$ and $F \in \mathbb{C}_\mathcal{D}$, we get that $\underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}} \preceq f(\mathbf{z})$ and $f(\mathbf{z}') \preceq \overline{S}_m(\mathbf{z}') - \overline{\mathbf{w}}$. Since f is monotone and $\mathbf{z} \preceq \mathbf{z}'$, we get $f(\mathbf{z}) \preceq f(\mathbf{z}')$. Finally, since $\mathbf{z} \in Z_1$ and $\mathbf{z}' \in Z_3$, we get from (8),
$$f^\star(\mathbf{z}) = \underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}} \preceq \overline{S}_m(\mathbf{z}') - \overline{\mathbf{w}} = f^\star(\mathbf{z}').$$

Hence, we established the monotonicity of function $f^\star$.

We now prove that the second requirement of Definition 2 is satisfied. Let us consider $k \in \mathbb{K}$, we already know that $\mathbf{z}_k \in Z_1 \cup Z_3$. In the case where $\mathbf{z}_k \in Z_1$, we have from (6) and (8),
$$\mathbf{y}_k \preceq \underline{S}_m(\mathbf{z}_k) - \underline{\mathbf{w}} + \overline{\mathbf{w}} = f^\star(\mathbf{z}_k) + \overline{\mathbf{w}}.$$
Moreover, since $S_m \in \mathbb{S}_\mathcal{D}$, we have
$$\mathbf{y}_k \succeq \underline{S}_m(\mathbf{z}_k) = f^\star(\mathbf{z}_k) + \underline{\mathbf{w}}.$$
Therefore, $\mathbf{y}_k \in F^\star(\mathbf{x}_k)$. A similar statement can be shown when $\mathbf{z}_k \in Z_3$. It follows that $F^\star \in \mathbb{C}_\mathcal{D}$.

Hence, we have proved that for all $\mathbf{z}^\star \in Z \setminus Z_0$ and for all $\mathbf{y}^\star \in S_m(\mathbf{z}^\star)$, there exists a map $F^\star \in \mathbb{C}_\mathcal{D}$ such that $\mathbf{y}^\star \in F^\star(\mathbf{z}^\star)$. Hence, for all map $S \in \mathbb{S}_\mathcal{D}$, we have $\mathbf{y}^\star \in S(\mathbf{z}^\star)$. Hence, S_m is a minimal simulating map. ■

We now show some useful properties of minimal simulating maps. The first one states that the minimal simulating map in (3) inherits the monotonicity property:

Proposition 2: The minimal simulating map S_m defined in (3) is monotone.

Proof: Let $\mathbf{z}, \mathbf{z}' \in Z$ such that $\mathbf{z} \preceq \mathbf{z}'$. Then, $\mathbb{K}^+(\mathbf{z}') \subseteq \mathbb{K}^+(\mathbf{z})$ and $\mathbb{K}^-(\mathbf{z}) \subseteq \mathbb{K}^-(\mathbf{z}')$ which give from (5) and (6), $\overline{S}_m(\mathbf{z}) \preceq \overline{S}_m(\mathbf{z}')$ and $\underline{S}_m(\mathbf{z}) \preceq \underline{S}_m(\mathbf{z}')$. Then, for all $\mathbf{y} \in S_m(\mathbf{z})$, $\mathbf{y} \preceq \overline{S}_m(\mathbf{z}) \preceq \mathbf{y}'$ with $\mathbf{y}' = \overline{S}_m(\mathbf{z}') \in S_m(\mathbf{z}')$. Similarly, we can show that for all $\mathbf{y}' \in S_m(\mathbf{z}')$, $\mathbf{y}' \succeq \underline{S}_m(\mathbf{z}') \succeq \mathbf{y}$ with $\mathbf{y} = \underline{S}_m(\mathbf{z}) \in S_m(\mathbf{z})$. Therefore, according to Definition 1, S_m is monotone. ■

The second result characterizes the minimal simulating map obtained from the fusion of two data sets:

Proposition 3: Given two data sets $\mathcal{D}', \mathcal{D}''$, consider the data set $\mathcal{D} = \mathcal{D}' \cup \mathcal{D}''$, and let S_m, S'_m, S''_m be the minimal simulating maps of $\mathcal{D}$, $\mathcal{D}'$, $\mathcal{D}''$ given by (3). Then, we have
$$\forall \mathbf{z} \in Z,\ S_m(\mathbf{z}) = S'_m(\mathbf{z}) \cap S''_m(\mathbf{z}).$$

Proof: The data set $\mathcal{D}$ can be written as
$$\mathcal{D} = \{(\mathbf{z}_k, \mathbf{y}_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), k \in \mathbb{K}_\mathcal{D}\},$$
with $\mathbb{K}_\mathcal{D} = \mathbb{K}_{\mathcal{D}'} \cup \mathbb{K}_{\mathcal{D}''}$ where $\mathbb{K}_{\mathcal{D}'}$ and $\mathbb{K}_{\mathcal{D}''}$ are the sets of indices of $\mathcal{D}'$ and $\mathcal{D}''$. Then, for all $\mathbf{z} \in Z$, we have
$$\mathbb{K}^-_\mathcal{D}(\mathbf{z}) = \mathbb{K}^-_{\mathcal{D}'}(\mathbf{z}) \cup \mathbb{K}^-_{\mathcal{D}''}(\mathbf{z}),\ \mathbb{K}^+_\mathcal{D}(\mathbf{z}) = \mathbb{K}^+_{\mathcal{D}'}(\mathbf{z}) \cup \mathbb{K}^+_{\mathcal{D}''}(\mathbf{z}).$$
Substituting in (3), we can easily show that for all $\mathbf{z} \in Z$ that $S_m(\mathbf{z}) = S'_m(\mathbf{z}) \cap S''_m(\mathbf{z})$. ■

Proposition 3 has several implications. The first one is that the computation of a minimal simulating map for large data sets can be easily parallelized. Indeed, one can split the data into multiple data sets, compute minimal simulating maps for these data sets concurrently, and then compute their intersection. The second implication is that pre-computed minimal simulating maps can be easily refined when new

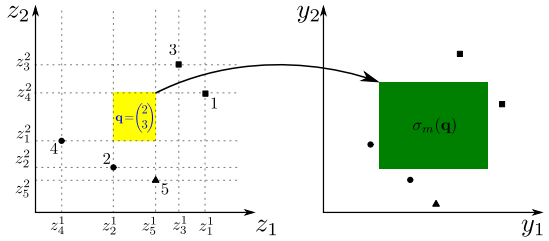

Fig. 5. The figure shows the rectangular partition of the input set $\boldsymbol{Z}$ based on the collected data points and an example of one of the discrete variables $\mathbf{q}$. All the points in yellow, aggregated in $\mathbf{q}$, have the same output.

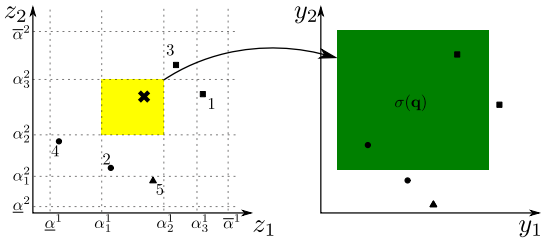

Fig. 6. The figure shows the predefined partition of the input set $\boldsymbol{Z}$. The function ϕ maps all the points inside a cell (e.g. square in yellow) to a discrete variable. The map $\boldsymbol{\sigma}$ is the minimal interval-valued simulating map.

data becomes available. Indeed, it is sufficient to compute a minimal simulating map for the new data and to intersect with the pre-computed one.

Remark 3: In this section, we provided a solution to Problem 1 for computing a minimal simulating map of $\mathcal{D}$, which provides a tight over-approximation of the map F. Note that similar results can be obtained if we seek an over-approximation of the function f as it can be shown that

$$\forall \mathbf{z} \in Z,\ f(\mathbf{z}) \in [\underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}}, \overline{S}_m(\mathbf{z}) - \overline{\mathbf{w}}],$$

which can be seen in Figure 2. If we want to calculate a single-valued approximation of the function f, we can use a similar approach to the one used in set membership technique [27] and build an estimation of f using the calculated bounds on the minimal simulating map

$$f(\mathbf{z}) = \frac{\overline{S}_m(\mathbf{z}) - \overline{\mathbf{w}} + \underline{S}_m(\mathbf{z}) - \underline{\mathbf{w}}}{2}.$$

It will be apparent from what follows that this estimation is piecewise constant and not continuous. Continuous estimation of f included in the over-approximation map S_m is derived in [22] using the class of multi-affine functions.

2) Computation: After characterizing minimal simulating maps, we move to address how to compute one in practice. Our approach and the used notations are illustrated in Figure 5, where a six data points case is demonstrated.

For that purpose, we introduce a rectangular partition of the input set Z, induced by the data $\mathcal{D}$. For simplicity, let us assume that Z is a closed interval of $\overline{\mathbb{R}}^n$, i.e. $Z = [\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}]$. This is without loss of generality since it is always possible to embed Z in such a set.

For each $i \in \{1, \dots, n\}$, we sort the i^{th} components of all the data points in the input space (i.e., z^i_k, for $k \in \mathbb{K}$), so we have $z^i_{k^1_i} \leq \cdots \leq z^i_{k^{|\mathbb{K}|}_i}$. The sorted values are then used to define the finite partitions $(Z^i_{q^i})_{q^i \in Q^i}$ of $[\underline{\alpha}^i, \overline{\alpha}^i]$ where $Q^i = \{0, \dots, |\mathbb{K}|\}$ and

$$\begin{cases} Z^i_0 &= [\underline{\alpha}^i, z^i_{k^1_i}), \\ Z^i_{q^i} &= [z^i_{k^{q^i}_i}, z^i_{k_i{}^{q^i+1}}),\ q^i = 1, \dots, |\mathbb{K}| - 1, \\ Z^i_{|\mathbb{K}|} &= [z^i_{k^{|\mathbb{K}|}_i}, \overline{\alpha}^i]. \end{cases}$$

Then, let us define $Q = Q^1 \times \cdots \times Q^n$, and let the finite rectangular partition $(Z_{\mathbf{q}})_{\mathbf{q} \in Q}$ of Z be given for $\mathbf{q} = (q^1, \dots, q^n)$ by $Z_{\mathbf{q}} = Z^1_{q^1} \times \cdots \times Z^n_{q^n}$.

Lemma 1: Let S_m be the minimal simulating map of $\mathcal{D}$ given by (3). Then, for all $\mathbf{q} \in Q$, for all $\mathbf{z}, \mathbf{z}' \in \operatorname{int} Z_{\mathbf{q}}$, $S_m(\mathbf{z}) = S_m(\mathbf{z}')$.

Proof: Because the partition is defined using the components of data points, we have, for all $\mathbf{q} \in Q$, for all $\mathbf{z}, \mathbf{z}' \in \operatorname{int} Z_{\mathbf{q}}$, $\mathbb{K}^-(\mathbf{z}) = \mathbb{K}^-(\mathbf{z}')$ and $\mathbb{K}^+(\mathbf{z}) = \mathbb{K}^+(\mathbf{z}')$. Subsequently, we obtain by (3), $S_m(\mathbf{z}) = S_m(\mathbf{z}')$. ■

Building on the property presented in Lemma 1, we can introduce the new map $\sigma_m : Q \rightrightarrows \overline{\mathbb{R}}^m$ defined as follows

$$\forall \mathbf{q} \in Q,\ \sigma_m(\mathbf{q}) = S_m(\mathbf{z}),\ \text{for } \mathbf{z} \in \operatorname{int} Z_{\mathbf{q}}. \tag{9}$$

We also define a quantization function $\phi_m : Z \to Q$ associated to the finite data-induced partition $(Z_{\mathbf{q}})_{\mathbf{q} \in Q}$ as follows

$$\forall \mathbf{z} \in Z,\ \forall \mathbf{q} \in Q,\ \phi_m(\mathbf{z}) = \mathbf{q} \iff \mathbf{z} \in Z_{\mathbf{q}}. \tag{10}$$

Proposition 4: Let σ_m and ϕ_m be given by (9) and (10), then $\sigma_m \circ \phi_m$ is a minimal simulating map of $\mathcal{D}$.

Proof: It follows directly from Lemma 1 and from the definition of σ_m and ϕ_m that $\sigma_m \circ \phi_m = S_m$ a.e.. Then, let us show that $\sigma_m \circ \phi_m$ is a simulation map. From the construction of the partition $(Z_{\mathbf{q}})_{\mathbf{q} \in Q}$, it follows that for all $\mathbf{q} \in Q$, for all $\mathbf{z} \in Z_{\mathbf{q}}$, for all $\mathbf{z}' \in \operatorname{int} Z_{\mathbf{q}}$, $\mathbb{K}^-(\mathbf{z}') \subseteq \mathbb{K}^-(\mathbf{z})$ and $\mathbb{K}^+(\mathbf{z}') \subseteq \mathbb{K}^+(\mathbf{z})$. Hence, from (3), $S_m(\mathbf{z}) \subseteq S_m(\mathbf{z}')$. Then, it results that $S_m(\mathbf{z}) \subseteq \sigma_m \circ \phi_m(\mathbf{z})$, for all $\mathbf{z} \in Z$. Since $S_m \in \mathbb{S}_{\mathcal{D}}$, we have that $\sigma_m \circ \phi_m \in \mathbb{S}_{\mathcal{D}}$. Therefore, according to Proposition 1, $\sigma_m \circ \phi_m$ is a minimal simulating map of $\mathcal{D}$. ■

Hence, computing the finite partition $(Z_{\mathbf{q}})_{\mathbf{q} \in Q}$ and the map σ_m offers an effective way to compute a minimal simulating map and store it. Let us remark that the number of elements in Q is $(|\mathbb{K}|+1)^n$. Hence, it grows polynomially with the number of data points $|\mathbb{K}|$ and exponentially with the dimension n of the input space Z. Therefore, it is computationally prohibitive to use this approach for large data sets, particularly with high-dimensional input spaces. In the following, we tackle this problem by fixing a partition a priori and by finding the best simulating map on this partition.

C. Simulating maps on fixed partitions

Instead of relying on the data set $\mathcal{D}$ to partition Z, which makes calculating the simulating map computationally expensive, we will assume that a rectangular partition of Z is given a priori. In the following, we characterize the tightest interval-valued simulating map on this partition. Our approach is illustrated in Figure 6

1) Construction: For each coordinate $i \in \{1,\dots,n\}$, let be given finite partitions $(D^i_{q^i})_{q^i \in Q^i}$ of $[\underline{\alpha}^i, \overline{\alpha}^i]$ where $Q^i = \{0,\dots,K^i\}$ and

$$\begin{cases} D^i_0 & = [\underline{\alpha}^i, \alpha^i_1), \\ D^i_{q^i} & = [\alpha^i_{q^i}, \alpha^i_{q^i+1}),\ q^i = 1,\dots,K^i-1, \\ D^i_{K^i} & = [\alpha^i_{K^i}, \overline{\alpha}^i], \end{cases}$$

where $\underline{\alpha}^i < \alpha^i_1 < \cdots < \alpha^i_{K^i} < \overline{\alpha}^i$. Note that the number of partition elements K^i+1 can be different from one coordinate to another. Then, let us define $Q = Q^1 \times \cdots \times Q^n$, and let the finite rectangular partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z be given for $\mathbf{q} = (q^1,\dots,q^n)$ by $D_{\mathbf{q}} = D^1_{q^1} \times \cdots \times D^n_{q^n}$.

Let us define the map $\sigma : Q \rightrightarrows \overline{\mathbb{R}}^m$ given for all $\mathbf{q} \in Q$ by

$$\begin{aligned} \sigma(\mathbf{q}) = &\left(\bigcap_{k\in\mathbb{K}^-(\underline{\mathbf{z}}_{\mathbf{q}})} \{\mathbf{y} \in \overline{\mathbb{R}}^m \,|\, \mathbf{y}_k + \underline{\mathbf{w}} - \overline{\mathbf{w}} \preceq \mathbf{y}\} \right) \\ &\cap \left(\bigcap_{k\in\mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})} \{\mathbf{y} \in \overline{\mathbb{R}}^m \,|\, \mathbf{y} \preceq \mathbf{y}_k + \overline{\mathbf{w}} - \underline{\mathbf{w}}\} \right) \end{aligned} \tag{11}$$

where $\underline{\mathbf{z}}_{\mathbf{q}} = \inf D_{\mathbf{q}}$, $\overline{\mathbf{z}}_{\mathbf{q}} = \sup D_{\mathbf{q}}$, and $\mathbb{K}^-$, $\mathbb{K}^+$ are defined as in (4). From (5), (6) and (11), we get that σ is an interval-valued map: for all $\mathbf{q} \in Q$, $\sigma(\mathbf{q}) = [\underline{\sigma}(\mathbf{q}), \overline{\sigma}(\mathbf{q})]$ with $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$ and $\underline{\sigma}(\mathbf{q}) = \underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}})$.

We also consider a quantization function $\phi : Z \to Q$ associated to the finite partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ and defined as

$$\forall \mathbf{z} \in Z,\ \forall \mathbf{q} \in Q,\ \phi(\mathbf{z}) = \mathbf{q} \iff \mathbf{z} \in D_{\mathbf{q}}. \tag{12}$$

Theorem 2: Let σ and ϕ be given by (11) and (12), then the following properties hold:

1) $\sigma \circ \phi \in \mathbb{S}_{\mathcal{D}}$ (Simulation);
2) For any interval-valued map $\sigma' : Q \rightrightarrows \overline{\mathbb{R}}^m$, such that $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$, it holds for all $\mathbf{q} \in Q$, $\sigma(\mathbf{q}) \subseteq \sigma'(\mathbf{q})$ (Minimality).

Proof: Let S_m be the minimal simulating map in (3). Let us prove the simulation property. Let $\mathbf{z} \in Z$, then $\underline{\mathbf{z}}_{\phi(\mathbf{z})} \preceq \mathbf{z} \preceq \overline{\mathbf{z}}_{\phi(\mathbf{z})}$, which yields

$$\mathbb{K}^-(\underline{\mathbf{z}}_{\phi(\mathbf{z})}) \subseteq \mathbb{K}^-(\mathbf{z}) \text{ and } \mathbb{K}^+(\overline{\mathbf{z}}_{\phi(\mathbf{z})}) \subseteq \mathbb{K}^+(\mathbf{z}).$$

Hence, by (3) and (11), $S_m(\mathbf{z}) \subseteq \sigma \circ \phi(\mathbf{z})$. Since $S_m \in \mathbb{S}_{\mathcal{D}}$, we get $\sigma \circ \phi \in \mathbb{S}_{\mathcal{D}}$.

Next, let us prove the minimality property. Let us consider an interval-valued map $\sigma' : Q \rightrightarrows \overline{\mathbb{R}}^m$ such that $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$, and let $q \in Q$. From (4), it can be seen that there exists a neighborhood $\overline{N}_{\mathbf{q}}$ of $\overline{\mathbf{z}}_{\mathbf{q}}$ such that for all $\mathbf{z} \in \overline{N}_{\mathbf{q}}$ with $\mathbf{z} \preceq \overline{\mathbf{z}}_{\mathbf{q}}$ it holds $\mathbb{K}^+(\mathbf{z}) = \mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})$, which yields by (3) $\overline{S}_m(\mathbf{z}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$. Then, let $\mathbf{z}^\star \in (\overline{N}_{\mathbf{q}} \cap D_{\mathbf{q}}) \setminus Z_0$ with Z_0 the set of measure zero defined as in (7). Let us remark that $\mathbf{z}^\star \preceq \overline{\mathbf{z}}_{\mathbf{q}}$ and hence $\overline{S}_m(\mathbf{z}^\star) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$. From the proof of Theorem 1, we get that there exists $F^\star \in \mathbb{C}_{\mathcal{D}}$ such that $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) = \overline{S}_m(\mathbf{z}^\star) \in F^\star(\mathbf{z}^\star)$. Then, $\sigma' \circ \phi \in \mathbb{S}_{\mathcal{D}}$ gives us that $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) \in \sigma' \circ \phi(\mathbf{z}^\star)$. Moreover, $\mathbf{z}^\star \in D_{\mathbf{q}}$, gives us that $\sigma' \circ \phi(\mathbf{z}^\star) = \sigma'(\mathbf{q})$. Hence, $\overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}}) \in \sigma'(\mathbf{q})$. Similarly, we can show that $\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}) \in \sigma'(\mathbf{q})$. Then since σ' is interval-valued, we get that $[\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}), \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})] \subseteq \sigma'(\mathbf{q})$. Then, $\sigma(\mathbf{q}) = [\underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}}), \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})]$ gives us $\sigma(\mathbf{q}) \subseteq \sigma'(\mathbf{q})$. ■

It follows from Theorem 2 that it is possible to define a notion of minimal simulating map of $\mathcal{D}$ relative to a given partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z. It should be noticed that similar results to Propositions 2 and 3 hold for the simulating map $\sigma \circ \phi$. Finally, one can check that if the partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z coincides with the data-induced partition defined in the previous section, $\sigma \circ \phi$ is minimal simulating map of $\mathcal{D}$. The difference between the maps $\sigma\circ\phi$ and $\sigma_m \circ \phi$ can be examined by comparing Figure 5 and Figure 6.

Remark 4: For a given number of cells, typically smaller than data-induced partition, choosing an optimal partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ is a complicated problem. However, it can be easily shown that the optimal partition (achieving the minimal over-approximation volume) is aligned with some of the data point components. This makes it possible to rely on heuristics to choose good yet sub-optimal partitions. For instance, a possible approach would be to aggregate cells of the data-induced partition where a lot of data is available.

2) Efficient computation: Given the partition $(D_{\mathbf{q}})_{\mathbf{q}\in Q}$ of Z, computing the simulating map defined in the previous section amounts to computing the map $\sigma : Q \rightrightarrows \overline{\mathbb{R}}^m$ given by (11). A straightforward algorithm to compute σ is as follows. For each $\mathbf{q} \in Q$, we go through all the data points in $\mathcal{D}$ and determine $\mathbb{K}^-(\underline{\mathbf{z}}_{\mathbf{q}})$ and $\mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}})$. Then, one gets $\sigma(\mathbf{q})$ by (11). That makes the overall complexity $\mathcal{O}(|\mathbb{K}| \times |Q|)$, i.e. bilinear with respect to the number of data points and to the number of the partition elements. In this section, we present a more efficient approach to calculating σ.

For simplicity, we assume in the following that for all $k \in \mathbb{K}$, $\mathbf{z}_k \in \operatorname{int} Z$, i.e. that no data point lies on the boundary of the input set. Then, since $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$, it follows from (5) that for all $\mathbf{q} \in Q$ such that $q^i = K^i$, for some $i \in \{1,\dots,n\}$, $\overline{\sigma}(\mathbf{q}) = (+\infty,\dots,+\infty)$. Similarly, since $\underline{\sigma}(\mathbf{q}) = \underline{S}_m(\underline{\mathbf{z}}_{\mathbf{q}})$, it follows from (6) that for all for all $\mathbf{q} \in Q$ such that $q^i = 0$, for some $i \in \{1,\dots,n\}$, $\underline{\sigma}(\mathbf{q}) = (-\infty,\dots,-\infty)$.

Then, let us consider the following subsets of Q:

$$\overline{Q} = \{0,\dots,K^1-1\} \times \cdots \times \{0,\dots,K^n-1\},$$
$$\underline{Q} = \{1,\dots,K^1\} \times \cdots \times \{1,\dots,K^n\}.$$

One needs to compute $\overline{\sigma}(\mathbf{q})$ and $\underline{\sigma}(\mathbf{q})$ for $\mathbf{q} \in \overline{Q}$ and $\mathbf{q} \in \underline{Q}$, respectively. For that purpose, let us first define the functions $\overline{\sigma}_0 : \overline{Q} \to \overline{\mathbb{R}}^m$ and $\underline{\sigma}_0 : \underline{Q} \to \overline{\mathbb{R}}^m$ as follows:

$$\overline{\sigma}_0(\mathbf{q}) = \inf \left\{ \mathbf{y}_k + \overline{\mathbf{w}} - \underline{\mathbf{w}} \mid \mathbf{z}_k \in \operatorname{cl} D_{\operatorname{up}(\mathbf{q})} \right\}, \tag{13}$$
$$\underline{\sigma}_0(\mathbf{q}) = \sup \left\{ \mathbf{y}_k + \underline{\mathbf{w}} - \overline{\mathbf{w}} \mid \mathbf{z}_k \in \operatorname{cl} D_{\operatorname{lo}(\mathbf{q})} \right\}, \tag{14}$$

where $\operatorname{up}(\mathbf{q}) = \mathbf{q} + \mathbb{1}_n$, $\operatorname{lo}(\mathbf{q}) = \mathbf{q} - \mathbb{1}_n$, and $\mathbb{1}_n = (1,\dots,1)$. An illustration of the elements $\operatorname{up}(\mathbf{q})$ and $\operatorname{lo}(\mathbf{q})$ can be seen in Figure 7. To compute $\overline{\sigma}_0, \underline{\sigma}_0$, we start by initializing $\overline{\sigma}_0(\mathbf{q}) = (+\infty,\dots,+\infty)$ for all $\mathbf{q} \in \overline{Q}$, and $\underline{\sigma}_0(\mathbf{q}) = (-\infty,\dots,-\infty)$ for all $\mathbf{q} \in \underline{Q}$. Then, we go through all the points in the set $\mathcal{D}$; for each entry $(\mathbf{z}_k, \mathbf{y}_k)$ we find all $\mathbf{q}$ such that $\mathbf{z}_k \in \operatorname{cl} D_{\mathbf{q}}$. Then, we update the value of $\underline{\sigma}_0(\operatorname{up}(\mathbf{q}))$ and $\overline{\sigma}_0(\operatorname{lo}(\mathbf{q}))$ using (13) and (14). The partition is stored and sorted component-wise, so to find $\mathbf{q}$, we can do a binary search for each component of $\mathbf{z}_k$. Therefore, computing $\overline{\sigma}_0, \underline{\sigma}_0$ is done with a complexity $\mathcal{O}(|\mathbb{K}| \times \sum_i \log(K^i))$ or equivalently $\mathcal{O}(|\mathbb{K}| \times \log(|Q|))$.

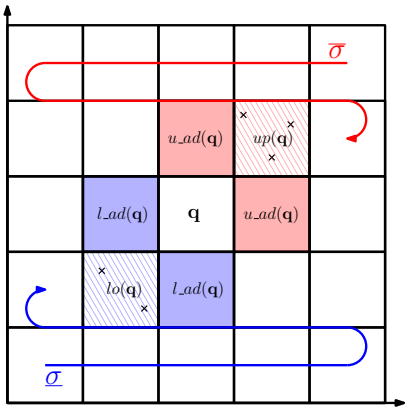

Fig. 7. A representation of the sets $\mathrm{up}(\mathbf{q})$, $\mathrm{u\_ad}(\mathbf{q})$, $\mathrm{lo}(\mathbf{q})$, $\mathrm{l\_ad}(\mathbf{q})$ and the order in which we calculate $\overline{\boldsymbol{\sigma}}$ and $\underline{\boldsymbol{\sigma}}$.

We now present a result that will allow us to compute the map σ sequentially.

Proposition 5: Let σ be the interval-valued map given by (11), its upper and lower bounds $\overline{\sigma}, \underline{\sigma}$ satisfy:

$$\forall \mathbf{q} \in \overline{Q},\ \overline{\sigma}(\mathbf{q}) = \min\left(\inf\left\{\overline{\sigma}(\mathbf{q}')\middle|\ \mathbf{q}' \in \mathrm{u\_ad}(\mathbf{q})\right\}, \overline{\sigma}_0(\mathbf{q})\right), \tag{15}$$

$$\forall \mathbf{q} \in \underline{Q},\ \underline{\sigma}(\mathbf{q}) = \max\left(\sup\left\{\underline{\sigma}(\mathbf{q}')\middle|\ \mathbf{q}' \in \mathrm{l\_ad}(\mathbf{q})\right\}, \underline{\sigma}_0(\mathbf{q})\right), \tag{16}$$

where

$$\mathrm{u\_ad}(\mathbf{q}) = \{\mathbf{q}' \in Q |\ \exists k \in \{1,\dots,n\}, \mathbf{q}' - \mathbf{q} = \mathbf{e}_k\},$$

$$\mathrm{l\_ad}(\mathbf{q}) = \{\mathbf{q}' \in Q |\ \exists k \in \{1,\dots,n\}, \mathbf{q} - \mathbf{q}' = \mathbf{e}_k\},$$

and $\mathbf{e}_k \in \mathbb{R}^n$ whose k^{th} component is 1 and all others are 0.

An illustration of the sets $\mathrm{u\_ad}(\mathbf{q})$ and $\mathrm{l\_ad}(\mathbf{q})$ can be seen in Figure 7.

Proof: We prove the property for the upper bound $\overline{\sigma}$; the proof for the lower bound $\underline{\sigma}$ follows similarly. Let $q \in \overline{Q}$, it can be seen that

$$\mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}}) = \left(\bigcup_{\mathbf{q}' \in \mathrm{u\_ad}(\mathbf{q})} \mathbb{K}^+(\overline{\mathbf{z}}_{\mathbf{q}'})\right) \cup \{k \in \mathbb{K} |\ \mathbf{z}_k \in D_{\mathrm{up}}(\mathbf{q})\}.$$

Then, since $\overline{\sigma}(\mathbf{q}) = \overline{S}_m(\overline{\mathbf{z}}_{\mathbf{q}})$, (15) follows directly from (5) and the equality above. ∎

From Proposition 5, we can see that to compute $\overline{\sigma}$ we go through all $\mathbf{q} \in Q$ sequentially in a decreasing order, starting from $\mathbf{q} = (K^1, \dots, K^n)$ as represented by Figure 7. For $\underline{\sigma}$ we start from $\mathbf{q} = (1, \dots, 1)$ and though all $\mathbf{q} \in Q$ in an increasing order.

Proposition 6: The map σ can be computed with complexity $\mathcal{O}(|\mathbb{K}| \times \log(|Q|) + |Q|)$.

Proof: We already showed the complexity of computing $\overline{\sigma}_0$ and $\underline{\sigma}_0$ is $\mathcal{O}(|\mathbb{K}| \times \log(|Q|))$. To compute σ we should go through all the elements $\mathbf{q} \in Q$ twice, one in decreasing order to compute $\overline{\sigma}$ and one in increasing order to compute $\underline{\sigma}$. Therefore, the complexity of computing σ is $\mathcal{O}(|\mathbb{K}| \times \log(|Q|) + |Q|)$. ∎

III. Data-Driven Abstraction

In this section, we show that our approach presented in the previous section can be used for data-driven modeling of discrete-time dynamical systems. We show that simulating maps computed from data can be used to define sound finite state abstractions for dynamical systems with unknown monotone dynamics. Then, we extend our approach to non-monotone systems with bounded derivatives.

We consider a discrete-time dynamical system:

$$\mathbf{x}(\tau + 1) \in f(\mathbf{x}(\tau), \mathbf{u}(\tau)) + g(\mathbf{x}(\tau), \mathbf{u}(\tau)) + W \tag{17}$$

where $\mathbf{x}(\tau) \in X \subseteq \overline{\mathbb{R}}^{n_x}$ and $\mathbf{u}(\tau) \in U \subseteq \overline{\mathbb{R}}^{n_u}$ denote the state and the control input, $g : X \times U \to \overline{\mathbb{R}}^{n_x}$ is a known function whereas $f : X \times U \to \overline{\mathbb{R}}^{n_x}$ is an unknown monotone function with respect to $\mathbf{z} = (\mathbf{x}, \mathbf{u})$, $W = [\underline{\mathbf{w}}, \overline{\mathbf{w}}] \subseteq \mathbb{R}^{n_x}$ is a bounded interval of disturbances with known bounds.

Let be given a set of data $\mathcal{D} \subseteq X \times U \times \overline{\mathbb{R}}^{n_x}$ consisting of transitions sampled from the system (17):

$$\mathcal{D} = \left\{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}'_k)\middle|\ \begin{array}{r}\mathbf{x}'_k \in f(\mathbf{x}_k, \mathbf{u}_k) + g(\mathbf{x}_k, \mathbf{u}_k) + W,\\ k \in \mathbb{K}\end{array}\right\} \tag{18}$$

where $\mathbb{K}$ is a finite set of indices. Because g is known, we can construct an auxiliary data set $\mathcal{D}'$ as follows:

$$\mathcal{D}' = \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{y}_k)|\ \mathbf{y}_k = \mathbf{x}'_k - g(\mathbf{x}_k, \mathbf{u}_k),\ k \in \mathbb{K}\}.$$

The new auxiliary data set can be seen as a set of data generated by a monotone map of the form (1).

Therefore, we can use the approach presented in the previous section to compute a simulating map $S : X \times U \rightrightarrows \overline{\mathbb{R}}^{n_x}$ of data $\mathcal{D}'$. Then, a data-driven model of system (17) can be defined as follows

$$\mathbf{x}(\tau + 1) \in S(\mathbf{x}(\tau), \mathbf{u}(\tau)) + g(\mathbf{x}(\tau), \mathbf{u}(\tau)). \tag{19}$$

In the following, we formally relate the behaviors of (17) and (19).

A. Alternating simulation relation

Alternating simulation [33] is a formal relationship between the behaviors of two systems that makes it possible to refine a controller synthesized for one system in order to control the other while preserving guarantees of correctness. Alternating simulation relations are usually defined within the framework of transition systems.

Definition 5: A *transition system* T is a tuple $T = (X, U, \Delta, Y, H)$, where X is a set of states, U is a set of inputs, $\Delta : X \times U \rightrightarrows X$ is a transition relation, Y is a set of outputs, and $H : X \to Y$ is an output map.

An input $\mathbf{u} \in U$ is called *enabled* at $\mathbf{x} \in X$ if $\Delta(\mathbf{x}, \mathbf{u}) \neq \emptyset$. The set of all inputs enabled at $\mathbf{x}$ is denoted $\mathrm{enab}_\Delta(\mathbf{x})$.

We define transition systems $T_{sys} = (X, U, \Delta_{sys}, Y, H)$ and $T_{data} = (X, U, \Delta_{sys}, Y, H)$ associated to (17) and (19) where the set of states X and inputs U are the same as in (17) and (19). The transition relation Δ_{sys} is defined as follows, for all $\mathbf{x} \in X$:

$$\mathbf{u} \in \mathrm{enab}_{\Delta_{sys}}(\mathbf{x}) \iff f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + W \subseteq X, \tag{20}$$

and

$$\forall \mathbf{u} \in \text{enab}_{\Delta_{sys}}(\mathbf{x}),\ \Delta_{sys}(\mathbf{x},\mathbf{u}) = f(\mathbf{x},\mathbf{u}) + g(\mathbf{x},\mathbf{u}) + W. \tag{21}$$

Similarly, the transition relation Δ_{data} is defined as follows, for all $\mathbf{x} \in X$:

$$\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x}) \iff S(\mathbf{x},\mathbf{u}) + g(\mathbf{x},\mathbf{u}) \subseteq X, \tag{22}$$

and

$$\forall \mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x}),\ \Delta_{data}(\mathbf{x},\mathbf{u}) = S(\mathbf{x},\mathbf{u}) + g(\mathbf{x},\mathbf{u}). \tag{23}$$

Let us remark that (20) and (22) ensure that an input $\mathbf{u}$ is enabled at state $\mathbf{x}$ only if it is guaranteed that the next state of (17) and (19) belongs to the set of states X. The set of outputs Y and the output map H are left unspecified. They can be chosen arbitrarily but are assumed to be the same for T_{sys} and T_{data}. One can, for instance, choose $Y = X$ and H to be the identity map.

In order to relate formally the behaviors of T_{sys} and T_{data}, we recall the notion of alternating simulation relation [33]:

Definition 6: Let us consider two transition systems $T_i = (X_i, U_i, \Delta_i, Y_i, H_i)$ $i = 1, 2$, sharing the same sets of outputs ($Y_1 = Y_2 = Y$). A relation $R \subseteq X_1 \times X_2$ is an *alternating simulation relation* from T_1 to T_2 if the following conditions are satisfied:

1) for all $\mathbf{x}_1 \in X_1$, there exists $\mathbf{x}_2 \in X_2$ such that $(\mathbf{x}_1, \mathbf{x}_2) \in R$;
2) for all $(\mathbf{x}_1, \mathbf{x}_2) \in R$, $H_1(\mathbf{x}_1) = H_2(\mathbf{x}_2)$;
3) for all $(\mathbf{x}_1, \mathbf{x}_2) \in R$, for all $\mathbf{u}_1 \in \text{enab}_{\Delta_1}(\mathbf{x}_1)$, there exists $\mathbf{u}_2 \in \text{enab}_{\Delta_2}(\mathbf{x}_2)$ such that for all $\mathbf{x}_2' \in \Delta_2(\mathbf{x}_2, \mathbf{u}_2)$, there exists $\mathbf{x}_1' \in \Delta_1(\mathbf{x}_1, \mathbf{u}_1)$ satisfying $(\mathbf{x}_1', \mathbf{x}_2') \in R$.

Then, we say that:

- T_1 is *alternatingly simulated* by T_2, denoted $T_1 \preceq_{\mathcal{AS}} T_2$, if there exists an alternating simulation relation R from T_1 to T_2;
- T_1 is *alternatingly bisimilar* to T_2, denoted $T_1 \cong_{\mathcal{AS}} T_2$, if there exists a relation R such that R is an alternating simulation relation from T_1 to T_2 and R^{-1} is an alternating simulation relation from T_2 to T_1.

Proposition 7: Let $S \in \mathbb{S}_{\mathcal{D}'}$, then for any choice of set of outputs Y and of output map $H : X \to Y$, $T_{data} \preceq_{\mathcal{AS}} T_{sys}$.

Proof: Let us show that

$$R = \{(\mathbf{x}_{data}, \mathbf{x}_{sys}) \in X \times X \mid \mathbf{x}_{data} = \mathbf{x}_{sys}\}$$

is an alternating simulation relation from T_{data} to T_{sys}. The first two conditions of alternating simulation follow directly from the form of R and from the fact that T_{sys} and T_{data} have the same sets of states X and of outputs Y, and the same output maps H.

Since $S \in \mathbb{S}_{\mathcal{D}'}$, it follows from Definition 3 that

$$\forall \mathbf{x} \in X, \mathbf{u} \in U,\ f(\mathbf{x},\mathbf{u}) + W \subseteq S(\mathbf{x},\mathbf{u}).$$

Hence, $S(\mathbf{x},\mathbf{u}) + g(\mathbf{x},\mathbf{u}) \subseteq X$ implies $f(\mathbf{x},\mathbf{u}) + g(\mathbf{x},\mathbf{u}) + W \subseteq X$. Therefore, from (20) and (22), we have for all $\mathbf{x} \in X$, $\text{enab}_{\Delta_{data}}(\mathbf{x}) \subseteq \text{enab}_{\Delta_{sys}}(\mathbf{x})$. Moreover, from (21) and (23), for all $\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x})$, it holds $\Delta_{sys}(\mathbf{x},\mathbf{u}) \subseteq \Delta_{data}(\mathbf{x},\mathbf{u})$.

Let us now show the third condition of alternating simulation. Let us consider $(\mathbf{x}_{sys}, \mathbf{x}_{data}) \in R$, then $\mathbf{x}_{sys} = \mathbf{x}_{data}$. Let $\mathbf{u}_{data} \in \text{enab}_{\Delta_{data}}(\mathbf{x}_{data})$, then for $\mathbf{u}_{sys} = \mathbf{u}_{data}$, we have $\mathbf{u}_{sys} \in \text{enab}_{\Delta_{sys}}(\mathbf{x}_{sys})$. Moreover, $\Delta_{sys}(\mathbf{x}_{sys}, \mathbf{u}_{sys}) \subseteq \Delta_{data}(\mathbf{x}_{data}, \mathbf{u}_{data})$. Therefore, for all $\mathbf{x}_{sys}' \in \Delta_{sys}(\mathbf{x}_{sys}, \mathbf{u}_{sys})$, there exists $\mathbf{x}_{data}' \in \Delta_{data}(\mathbf{x}_{data}, \mathbf{u}_{data})$ satisfying $\mathbf{x}_{sys}' = \mathbf{x}_{data}'$, and hence $(\mathbf{x}_{sys}', \mathbf{x}_{data}') \in R$. ■

From the previous result, it follows that the data-driven model T_{data} can be used to synthesize controllers that can be refined into controllers for the partially unknown system T_{sys}, with formal guarantees of correctness.

B. Symbolic abstraction

We now go one step further by computing symbolic abstractions of T_{data}. This will allow us to use discrete controller synthesis techniques to control the system with formal guarantees on the closed-loop behavior. Let us assume that the sets of states and inputs X and U are closed intervals of $\overline{\mathbb{R}}^{n_x}$ and $\overline{\mathbb{R}}^{n_u}$ and let $(X_{\mathbf{q}})_{\mathbf{q}\in Q}$, $(U_{\mathbf{p}})_{\mathbf{p}\in P}$ be given rectangular partitions of X and U as defined for Z in Section II-C. We define a quantization function $\phi : X \times U \to Q \times P$ associated to these finite partitions as follows,

$$\forall (\mathbf{x},\mathbf{u}) \in X \times U,\ \forall (\mathbf{q},\mathbf{p}) \in Q \times P,$$
$$\phi(\mathbf{x},\mathbf{u}) = (\mathbf{q},\mathbf{p}) \iff \mathbf{x} \in X_{\mathbf{q}},\ \mathbf{u} \in U_{\mathbf{p}}. \tag{24}$$

Then, we can use the approach presented in Section II-C to compute a map $\sigma : Q \times P \rightrightarrows \overline{\mathbb{R}}^{n_x}$ such that $\sigma \circ \phi$ is a simulating map of $\mathcal{D}'$.

Let us assume that for all $(\mathbf{q},\mathbf{p}) \in Q \times P$, we can compute subsets $G(\mathbf{q},\mathbf{p}) \subseteq \overline{\mathbb{R}}^{n_x}$ such that

$$\forall \mathbf{x} \in X_{\mathbf{q}},\ \forall\, \mathbf{u} \in U_{\mathbf{p}},\ g(\mathbf{x},\mathbf{u}) \in G(\mathbf{q},\mathbf{p}). \tag{25}$$

Such sets can be computed, for instance, using interval analysis [14] or using approaches based on mixed monotonicity or on growth bounds [23].

We define a symbolic transition system $T_{symb} = (Q, P, \Delta_{symb}, Y, H_{symb})$ where the set of states and inputs are given by the partitions index sets Q and P, and the transition relation Δ_{symb} is defined as follows, for all $\mathbf{q} \in Q$:

$$\mathbf{p} \in \text{enab}_{\Delta_{symb}}(\mathbf{q}) \iff \sigma(\mathbf{q},\mathbf{p}) + G(\mathbf{q},\mathbf{p}) \subseteq X, \tag{26}$$

and

$$\forall \mathbf{p} \in \text{enab}_{\Delta_{symb}}(\mathbf{q}),$$
$$\Delta_{symb}(\mathbf{q},\mathbf{p}) = \left\{\mathbf{q}' \in Q \middle|\ (\sigma(\mathbf{q},\mathbf{p}) + G(\mathbf{q},\mathbf{p})) \cap X_{\mathbf{q}}' \neq \emptyset\right\}. \tag{27}$$

We define the set of outputs to be $Y = Q$ and the output map H_{symb} to be the identity map.

Theorem 3: Let $S = \sigma \circ \phi \in \mathbb{S}_{\mathcal{D}'}$, let T_{sys} and T_{data} be defined as Section III-A for the set of outputs $Y = Q$ and the output map $H : X \to Q$, given by

$$\forall \mathbf{x} \in X,\ \forall \mathbf{q} \in Q,\ H(\mathbf{x}) = \mathbf{q} \iff \mathbf{x} \in X_{\mathbf{q}}. \tag{28}$$

Then, the following relation holds:

1) $T_{symb} \preceq_{\mathcal{AS}} T_{data} \preceq_{\mathcal{AS}} T_{sys}$.

2) If $g = 0$, then $T_{symb} \cong_{\mathcal{AS}} T_{data}$.

Proof: The fact that $T_{data} \preceq_{\mathcal{AS}} T_{sys}$ follows directly from Proposition 7. Then, let us show that

$$R = \{(\mathbf{q}, \mathbf{x}) \in Q \times X \mid \mathbf{x} \in X_{\mathbf{q}}\}$$

is an alternating simulation relation from T_{symb} to T_{data}. Let $\mathbf{q} \in Q$ and let $\mathbf{x} \in X_{\mathbf{q}}$, then $(\mathbf{q}, \mathbf{x}) \in R$ and the first condition of alternating simulation holds. Let $(\mathbf{q}, \mathbf{x}) \in R$, then $\mathbf{x} \in X_{\mathbf{q}}$, which by (28) gives $H(\mathbf{x}) = \mathbf{q}$. Since $H_{symb}(\mathbf{q}) = \mathbf{q}$, the second condition of alternating simulation holds.

Let us now show the third condition of alternating simulation. Let $(\mathbf{q}, \mathbf{x}) \in R$ and $\mathbf{p} \in \text{enab}_{\Delta_{symb}}(\mathbf{q})$, then choose $\mathbf{u} \in U_{\mathbf{p}}$. Since $S = \sigma \circ \phi$, we have $S(\mathbf{x}, \mathbf{u}) = \sigma(\mathbf{q}, \mathbf{p})$, and by (25) we have $g(\mathbf{x}, \mathbf{u}) \in G(\mathbf{q}, \mathbf{p})$. Hence,

$$S(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) \subseteq \sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p}). \tag{29}$$

Then, it follows from (22) and (26) that $\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x})$. Let $\mathbf{x}' \in \Delta_{data}(\mathbf{x}, \mathbf{u})$, from (23) and (29), we get $\mathbf{x}' \in \sigma(\mathbf{q}, \mathbf{p}) + G(\mathbf{q}, \mathbf{p})$. Let $\mathbf{q}' \in Q$ such that $\mathbf{x}' \in X_{\mathbf{q}'}$, from (27), we get that $\mathbf{q}' \in \Delta_{symb}(\mathbf{q}, \mathbf{p})$. Moreover, since $\mathbf{x}' \in X_{\mathbf{q}'}$, $(\mathbf{q}', \mathbf{x}') \in R$. Hence, we proved that $T_{symb} \preceq_{\mathcal{AS}} T_{data}$.

Now, let us assume that for all $\mathbf{x} \in X$, and $\mathbf{u} \in U$, $g(\mathbf{x}, \mathbf{u}) = 0$. Then, (25) holds with $G(\mathbf{q}, \mathbf{p}) = \{0\}$, for all $\mathbf{q} \in Q$ and $\mathbf{p} \in P$. Let us show that in that case, R^{-1} is an alternating simulation relation from T_{data} to T_{symb}. Let $\mathbf{x} \in X$, since $(X_{\mathbf{q}})_{\mathbf{q} \in Q}$ is a partition of X, there exists $\mathbf{q} \in Q$ such that $\mathbf{x} \in X_{\mathbf{q}}$. Hence, $(\mathbf{x}, \mathbf{q}) \in R^{-1}$ and the first condition of alternating simulation holds. The proof that the second condition holds for R^{-1} is the same as for R.

Then, let us show the third condition of alternating simulation. Let $(\mathbf{x}, \mathbf{q}) \in R^{-1}$ and $\mathbf{u} \in \text{enab}_{\Delta_{data}}(\mathbf{x})$, then there exists $\mathbf{p}$ such that $\mathbf{u} \in U_{\mathbf{p}}$. Since $S = \sigma \circ \phi$, we have $S(\mathbf{x}, \mathbf{u}) = \sigma(\mathbf{q}, \mathbf{p})$. Moreover, since $g(\mathbf{x}, \mathbf{u}) = 0$ and $G(\mathbf{q}, \mathbf{p}) = \{0\}$, we get from (22) and (26) that $\mathbf{p} \in \text{enab}_{\Delta_{symb}}(\mathbf{q})$. Let $\mathbf{q}' \in \Delta_{symb}(\mathbf{q}, \mathbf{p})$, from (27) we get that $\sigma(\mathbf{q}, \mathbf{p}) \cap X_{\mathbf{q}'} \neq \emptyset$. Hence, $S(\mathbf{x}, \mathbf{u}) \cap X_{\mathbf{q}'} \neq \emptyset$. Then, let $\mathbf{x}' \in S(\mathbf{x}, \mathbf{u}) \cap X_{\mathbf{q}'}$. From (23), $\mathbf{x}' \in \Delta_{data}(\mathbf{x}, \mathbf{u})$. Moreover $(\mathbf{x}', \mathbf{q}') \in R^{-1}$. Hence, we proved that $T_{data} \cong_{\mathcal{AS}} T_{symb}$. ∎

Theorem 3 shows that the symbolic abstraction T_{symb} can be used to synthesize controllers for the data-driven system T_{data} and hence also for the partially unknown system T_{sys}. Moreover, when T_{sys} is fully unknown and monotone (i.e. when $g = 0$), then Theorem 3 shows that working with the symbolic abstraction T_{symb} does not bring any conservatism compared to working with the data-driven model T_{data}.

Remark 5: The fact that T_{symb} has only a finite number of state and input values allows us to use algorithmic techniques to synthesize controllers for various specifications such as safety, reachability, or attractivity [8] or even more complex specifications expressed e.g. in linear temporal logic [1] or using hybrid automata [32]. Due to the fact that there is an alternating simulation relation between T_{sys} and T_{symb}, the synthesized controller can be refined such that the desired closed-loop behavior of T_{sys} is guaranteed to be correct-by-design (see e.g. [33] for more details on discrete controller refinement).

C. Systems with bounded derivatives

We now show how our approach can be adapted to compute data-driven models for systems of the form (17) where the unknown dynamics f is not necessarily monotone but has bounded derivatives with known upper and lower bounds. Our construction is inspired by the approach presented in [34] for computing decomposition functions of mixed-monotone functions.

Let us assume that for all $\mathbf{x} \in X$, $\mathbf{u} \in U$:

$$\frac{\partial f^i}{\partial x^j}(\mathbf{x}, \mathbf{u}) \in [\underline{a}_{ij}, \overline{a}_{ij}],\ i, j \in \{1, \dots, n_x\},$$
$$\frac{\partial f^i}{\partial u^j}(\mathbf{x}, \mathbf{u}) \in [\underline{b}_{ij}, \overline{b}_{ij}],\ i \in \{1, \dots, n_x\},\ j \in \{1, \dots, n_u\}.$$

where the bounds $\underline{a}_{ij}, \overline{a}_{ij}, \underline{b}_{ij}, \overline{b}_{ij} \in \mathbb{R}$ are known. Let us introduce the auxiliary matrices $A^-, A^+ \in \mathbb{R}^{n_x \times n_x}$ and $B^-, B^+ \in \mathbb{R}^{n_x \times n_u}$, where for all $i, j \in \{1, \dots, n_x\}$

$$A^-_{ij} = \begin{cases} \underline{a}_{ij} & \text{if } \underline{a}_{ij} < 0, \\ 0 & \text{otherwise,} \end{cases} \qquad A^+_{ij} = \begin{cases} \overline{a}_{ij} & \text{if } \overline{a}_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

and for all $i \in \{1, \dots, n_x\}$, for all $j \in \{1, \dots, n_u\}$

$$B^-_{ij} = \begin{cases} \underline{b}_{ij} & \text{if } \underline{b}_{ij} < 0, \\ 0 & \text{otherwise,} \end{cases} \qquad B^+_{ij} = \begin{cases} \overline{b}_{ij} & \text{if } \overline{b}_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Then, let the functions $f^-, f^+ : X \times U \to \overline{\mathbb{R}}^{n_x}$ and $g^-, g^+ : X \times U \to \overline{\mathbb{R}}^{n_u}$ be defined for all $\mathbf{x} \in X$, $\mathbf{u} \in U$, by:

$$f^-(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u}) - A^-\mathbf{x} - B^-\mathbf{u},$$
$$f^+(\mathbf{x}, \mathbf{u}) = A^+\mathbf{x} - B^+\mathbf{u} - f(\mathbf{x}, \mathbf{u}),$$
$$g^-(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + A^-\mathbf{x} + B^-\mathbf{u},$$
$$g^+(\mathbf{x}, \mathbf{u}) = g(\mathbf{x}, \mathbf{u}) + A^+\mathbf{x} + B^+\mathbf{u}.$$

Let us remark that g^-, g^+ are known, while f^-, f^+ are unknown but monotone since it can be readily checked that all their partial derivatives are nonnegative.

Given a data set $\mathcal{D}$ as in (18), we can define two auxiliary data sets:

$$\mathcal{D}^- = \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{y}^-_k) |\ \mathbf{y}^-_k = \mathbf{x}'_k - g^-(\mathbf{x}_k, \mathbf{u}_k),\ k \in \mathbb{K}\},$$
$$\mathcal{D}^+ = \{(\mathbf{x}_k, \mathbf{u}_k, \mathbf{y}^+_k) |\ \mathbf{y}^+_k = g^+(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}'_k,\ k \in \mathbb{K}\}.$$

We can use the approach presented in Section II to compute simulating maps $S^-, S^+ : X \times U \rightrightarrows \overline{\mathbb{R}}^{n_x}$ of data $\mathcal{D}^-$ and $\mathcal{D}^+$, respectively.

Proposition 8: Let $S^- \in \mathbb{S}_{\mathcal{D}^-}$ and $S^+ \in \mathbb{S}_{\mathcal{D}^+}$, then let $S : X \times U \rightrightarrows \overline{\mathbb{R}}^{n_x}$ be given for all $\mathbf{x} \in X$, $\mathbf{u} \in U$ by:

$$S(\mathbf{x}, \mathbf{u}) = \left(g^-(\mathbf{x}, \mathbf{u}) + S^-(\mathbf{x}, \mathbf{u})\right) \cap \left(g^+(\mathbf{x}, \mathbf{u}) - S^+(\mathbf{x}, \mathbf{u})\right).$$

Then, it holds:

$$\forall \mathbf{x} \in X, \mathbf{u} \in U,\ f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + W \subseteq S(\mathbf{x}, \mathbf{u}).$$

Proof: Let $\mathbf{x} \in X$, $\mathbf{u} \in U$, from $S^- \in \mathbb{S}_{\mathcal{D}^-}$, we have

$$f^-(\mathbf{x}, \mathbf{u}) + W \subseteq S^-(\mathbf{x}, \mathbf{u}).$$

Since $f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) = f^-(\mathbf{x}, \mathbf{u}) + g^-(\mathbf{x}, \mathbf{u})$, we get

$$f(\mathbf{x}, \mathbf{u}) + g(\mathbf{x}, \mathbf{u}) + W \subseteq g^-(\mathbf{x}, \mathbf{u}) + S^-(\mathbf{x}, \mathbf{u}).$$

Similarly, we can show that

$$f(\mathbf{x},\mathbf{u})+g(\mathbf{x},\mathbf{u})+W \subseteq g^+(\mathbf{x},\mathbf{u})-S^+(\mathbf{x},\mathbf{u}),$$

which leads to the result of the proposition. ∎

From Proposition 8, a data-driven model of system (17) can be defined as follows

$$\mathbf{x}(\tau+1) \in S(\mathbf{x}(\tau),\mathbf{u}(\tau)). \tag{30}$$

Following similar approaches to that described in Sections III-A and III-B, one can define a data-driven transition system T_{data} and a symbolic abstraction T_{symb} such that $T_{symb} \preceq_{\mathcal{AS}} T_{data} \preceq_{\mathcal{AS}} T_{sys}$.

IV. Data-Driven Verification of Assumptions

In the previous sections, we made several assumptions on the unknown map F: bounds on the disturbances and bounds on the partial derivatives of the function f. It is sometimes possible to derive such bounds a priori. For instance, in some cases, the monotonicity of a function can be inferred from principles of physics. However, there are many situations where these bounds also need to be inferred from data. In this section, we briefly present an approach to compute bounds on the disturbances and on the partial derivatives of the function f directly from data, with probabilistic guarantees. Our approach is based on the scenario approach [3], a data-driven approach to robust convex optimization.

Let us consider a map $F : Z \rightrightarrows \overline{\mathbb{R}}^m$ where $Z \subseteq \overline{\mathbb{R}}^n$ and such that for all $\mathbf{z} \in Z$, $F(\mathbf{z}) = f(\mathbf{z})+W$ where $f : Z \to \overline{\mathbb{R}}^m$ is a differentiable function with unknown lower and upper bounds $\underline{a}_{ij}, \overline{a}_{ij}$ on its partial derivatives $\frac{\partial f^i}{\partial z^j}$ and W, $W \subseteq \mathbb{R}^m$ is a bounded interval of disturbances with unknown lower and upper bounds $\underline{\mathbf{w}}, \overline{\mathbf{w}}$. Let us remark that without loss of generality, it is always possible to choose $\underline{\mathbf{w}} = -\overline{\mathbf{w}}$.

For all $i \in \{1,\dots,m\}$, let $\underline{\mathbf{a}}_i = (\underline{a}_{i1},\dots,\underline{a}_{in})$ and $\overline{\mathbf{a}}_i = (\overline{a}_{i1},\dots,\overline{a}_{in})$. Some bounds are consistent with our assumptions if and only if they satisfy:

$$\begin{aligned}&\forall \mathbf{z},\mathbf{z}' \in Z,\ \forall \mathbf{y} \in F(\mathbf{z}), \forall \mathbf{y}' \in F(\mathbf{z}'),\\ &\quad y^i - y'^i \le \overline{\mathbf{a}}_i \cdot [\mathbf{z}-\mathbf{z}']^+ + \underline{\mathbf{a}}_i \cdot [\mathbf{z}-\mathbf{z}']^- + 2\overline{w}^i\end{aligned} \tag{31}$$

where $[\mathbf{z}]^+ = \max(\mathbf{z},0)$ and $[\mathbf{z}]^- = \min(\mathbf{z},0)$.

Let us assume that we are given a set of random data generated by the map F:

$$\tilde{\mathcal{D}} = \{(\mathbf{z}_k, \mathbf{y}_k, \mathbf{z}'_k, \mathbf{y}'_k) \mid \mathbf{y}_k \in F(\mathbf{z}_k), \mathbf{y}'_k \in F(\mathbf{z}'_k), k \in \tilde{\mathbb{K}}\}$$

where $\tilde{\mathbb{K}}$ is a finite set of indices. We assume that the samples in $\tilde{\mathcal{D}}$ are independent and identically distributed. We aim at computing bounds such that (31) holds with probabilistic guarantees. This can be done using the scenario approach [3], which essentially consists in computing bounds such that the inequality in (31) holds at all points in $\tilde{\mathcal{D}}$. However, to obtain high confidence bounds, one needs to consider large data sets, resulting in large linear programs that can be complicated to solve in practice. For that reason, we present the following two-step approach that allows us to deal with very large data sets in practice. Let us partition the set of $\tilde{\mathcal{D}}$ in two subsets $\tilde{\mathcal{D}}'$ and $\tilde{\mathcal{D}}''$ indexed by $\tilde{\mathbb{K}}'$ and $\tilde{\mathbb{K}}''$ such that $\tilde{\mathbb{K}} = \tilde{\mathbb{K}}' \cup \tilde{\mathbb{K}}''$. In the first step, $\tilde{\mathcal{D}}'$ will be used to estimate the bounds $\underline{a}_{ij}, \overline{a}_{ij}$. In the second step, $\tilde{\mathcal{D}}''$ will be used to estimate the bound $\overline{w}^i$. Typically, the number of samples in $\tilde{\mathcal{D}}''$ will be much larger than that in $\tilde{\mathcal{D}}'$.

We first consider the following linear program:

$$\begin{cases}\min\limits_{\underline{\mathbf{a}}_i,\overline{\mathbf{a}}_i,\overline{w}^i} & \overline{\mathbf{a}}_i \cdot \sum_{k\in\tilde{\mathbb{K}}'}[\mathbf{z}_k-\mathbf{z}'_k]^+ + \underline{\mathbf{a}}_i \cdot \sum_{k\in\tilde{\mathbb{K}}'}[\mathbf{z}_k-\mathbf{z}'_k]^- + 2|\tilde{\mathbb{K}}'|\overline{w}^i\\ \text{s.t.} & y^i_k - y'^i_k \le \overline{\mathbf{a}}_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^+\\ & \qquad\qquad + \underline{\mathbf{a}}_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^- + 2\overline{w}^i,\ \forall k \in \tilde{\mathbb{K}}'\\ & \underline{\mathbf{a}}_i \preceq \overline{\mathbf{a}}_i.\end{cases} \tag{32}$$

The constraints of (32) are chosen such that (31) holds at all points in $\tilde{\mathcal{D}}'$, while the cost is chosen so as to minimize the average value of the right-hand side of (31). Let $\underline{\mathbf{a}}^\star_i, \overline{\mathbf{a}}^\star_i$ denote the optimal values of $\underline{\mathbf{a}}_i, \overline{\mathbf{a}}_i$ in (32). The optimal value of $\overline{w}^i$ is disregarded and estimated again in the second step.

Indeed, let us define

$$\overline{w}^{\star i} = \frac{1}{2}\max\left\{ y^i_k - y'^i_k - \overline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^+ - \underline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^- \;\middle|\; k \in \tilde{\mathbb{K}}'' \right\}. \tag{33}$$

Let us remark that while (32) can be solved using a moderately large data set $\tilde{\mathcal{D}}'$, (33) can easily be estimated on very large data sets. Moreover, we can provide probabilistic guarantees on the estimated bounds:

Proposition 9: Let $\beta \in (0,1)$ be a confidence parameter and let $i \in \{1,\dots,m\}$, let $\underline{\mathbf{a}}^\star_i, \overline{\mathbf{a}}^\star_i$ and $\overline{w}^{\star i}$ be estimated bounds using (32) and (33), respectively. Then, with probability at least $1-\beta$, it holds that

$$\mathbb{P}(y^i - y'^i \le \overline{\mathbf{a}}^\star_i \cdot [\mathbf{z}-\mathbf{z}']^+ + \underline{\mathbf{a}}^\star_i \cdot [\mathbf{z}-\mathbf{z}']^- + 2\overline{w}^{\star i}) \ge \beta^{\frac{1}{|\tilde{\mathbb{K}}''|}}. \tag{34}$$

Proof: Consider the following robust linear program

$$\begin{cases}\min\limits_{\overline{w}^i} & \overline{w}^i\\ \text{s.t.} & y^i - y'^i \le \overline{\mathbf{a}}^\star_i \cdot [\mathbf{z}-\mathbf{z}']^+ + \underline{\mathbf{a}}^\star_i \cdot [\mathbf{z}-\mathbf{z}']^- + 2\overline{w}^i,\\ & \qquad\qquad \forall \mathbf{z},\mathbf{z}' \in Z,\ \forall \mathbf{y} \in F(\mathbf{z}), \forall \mathbf{y}' \in F(\mathbf{z}').\end{cases} \tag{35}$$

Following the scenario approach, we consider the following associated scenario linear program:

$$\begin{cases}\min\limits_{\overline{w}^i} & \overline{w}^i\\ \text{s.t.} & y^i_k - y'^i_k \le \overline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^+\\ & \qquad\qquad + \underline{\mathbf{a}}^\star_i \cdot [\mathbf{z}_k-\mathbf{z}'_k]^- + 2\overline{w}^i,\ \forall k \in \tilde{\mathbb{K}}''.\end{cases} \tag{36}$$

It is easy to see that (33) provides the unique solution to (36). Moreover, from the results in [3], we get that (34) holds for the solution of (36). ∎

Hence, we have provided a method to estimate bounds on the disturbances and bounds on the partial derivatives. By using a two-step approach, our method can be used with very large data sets and can thus estimate the bounds with very high confidence. Let us remark that the method can be used to check if the system is monotone by checking if $\underline{\mathbf{a}}^\star_i \succeq 0$, for all $i \in \{1,\dots,m\}$.

V. Numerical Examples

In this section, we present three numerical examples. In the first, we test the performance of the introduced over-approximation. In the second, we study the case of a system with an unknown monotone part. Finally, we finish by showing the case of a system with bounded derivatives.

For all of the examples, we use interval domains, $Z = [\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}}]$, $\underline{\boldsymbol{\alpha}}, \overline{\boldsymbol{\alpha}} \in \overline{\mathbb{R}}^n$ for the maps we are trying to over-approximate. For $i \in \{1, \dots, n\}$, we define the finite partition $(D^i_{q^i})_{q^i \in Q^i}$ as follows, $Q^i = \{0, \dots, K^i\}$ and

$$\begin{cases} D^i_0 & = [\underline{\alpha}^i, \alpha^i_1) \\ D^i_{q^i} & = [\frac{q^i-1}{K^i-1}(\alpha^i_{K^i} - \alpha^i_1) + \alpha^i_1, \frac{q^i}{K^i-1}(\alpha^i_{K^i} - \alpha^i_1) + \alpha^i_1), \\ & \qquad q^i = 1, \dots, K^i - 1 \\ D^i_{K^i} & = [\alpha^i_{K^i}, \overline{\alpha}^i] \end{cases}$$

For the three examples, we choose $\alpha^i_1 = \underline{\alpha}^i + c\ (\overline{\alpha}^i - \underline{\alpha}^i)$, $\alpha^i_{K^i} = \overline{\alpha}^i - c\ (\overline{\alpha}^i - \underline{\alpha}^i)$, and c is a constant specific to each example.

A. Over-approximating an unknown function

In the first example, we present a set of experiments envisioned to test and visualize the algorithms introduced in this paper for the over-approximation of set-valued maps.

To quantitatively measure the performance of the over-approximation, we can check the execution time and the conservatism in the resulting over-approximation. We calculate the conservatism of the over-approximation using the following performance criterion

$$\mu(\mathcal{D}, Q) = \frac{\sum_{\mathbf{q} \in Q'} (vol(Z_{\mathbf{q}}) \times \sigma(\mathbf{q}))}{\sum_{\mathbf{q} \in Q'} (vol(Z_{\mathbf{q}}) \times vol(W))}$$

where

$$Q' = \{\mathbf{q} \in Q \big| \ -\infty < \underline{\sigma}^i(\mathbf{q}),\ \overline{\sigma}^i(\mathbf{q}) < \infty, \forall i \in \{1, \dots, n\}\}.$$

The denominator of μ represents the volume of the graph of the unknown map for the part of space where we can find an over-approximation, whereas the numerator represents the volume of our over-approximation. μ can take its value in the interval $[1, \infty)$, and the smaller its value is, the less conservative the over-approximation is.

In this example, we consider a monotone set-valued map $F : [-\pi, \pi] \times [-\pi, \pi] \rightrightarrows \overline{\mathbb{R}}$ given by

$$F(\mathbf{z}) = \{2\ (\sin z^1 + \sin z^2 + z^1 + z^2) + w \mid w \in [-0.1, 0.1]\} \tag{37}$$

We use F to generate the sets of data $\mathcal{D}$ used in the subsequent experiments. First, we visualized the over-approximation. We sampled $|\mathbb{K}| = 10^6$ data points. The parameters of the partition are chosen as follows $K^1 = 30$, $K^2 = 30$, $c = 0.01$. Figure 8 shows the undisturbed function in solid and the over-approximation calculated from data. We see how the undisturbed function is included in the over-approximation.

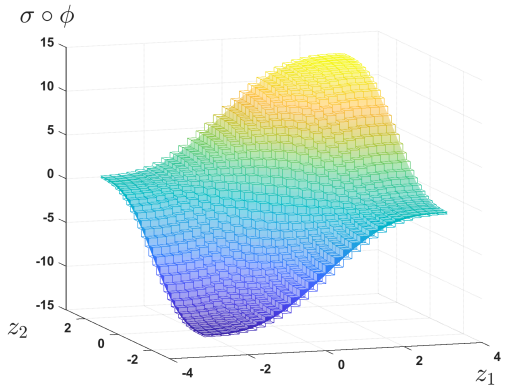

Fig. 8. Map $\boldsymbol{F(z)}$ with $\boldsymbol{w = 0}$ everywhere is represented in solid. The upper and lower bounds of the over-approximation are represented in transparency

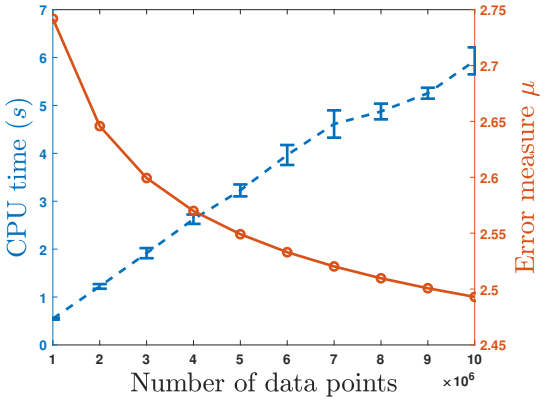

Fig. 9. Line in blue represents the average time of execution with respect to the number of data points, with bars representing the standard deviation. Line in orange shows the relation between the number of points and the performance criterion $\boldsymbol{\mu(\mathcal{D}, Q)}$.

1) The effect of changing the number of data points: To study the effect of changing the number of data points, we chose and fixed a partition, $K^1 = 100$, $K^2 = 100$, $c = 0.01$. Then, for an increasing number of data points, we calculated the over-approximation of the map F and measured the execution time and the performance criterion. For each number of data points, we redo the experiment a hundred times using different randomly generated data sets. The results of this statistical study of changing the number of data points are shown in Figure 9. We can see from the figure the linear relation between the number of data points and the execution time as predicted in Proposition 6. We also see how the conservatism in the calculated over-approximation decreases with the increase in the number of data points. Note that even if the number of points increases toward infinity, the performance criterion μ will not reach one because we are using a fixed partition. For μ to reach one, both the number of data points and the number of partition elements should go to infinity.

2) The effect of changing the size of the partition: In the second experiment, we fixed the number of data points $|\mathbb{K}| = 10^6$, and changed the size of the partition. For each size considered, the partition is chosen such that $K^1 = K^2$, and $c = 0.01$. We also redo the calculation for each size a hundred times

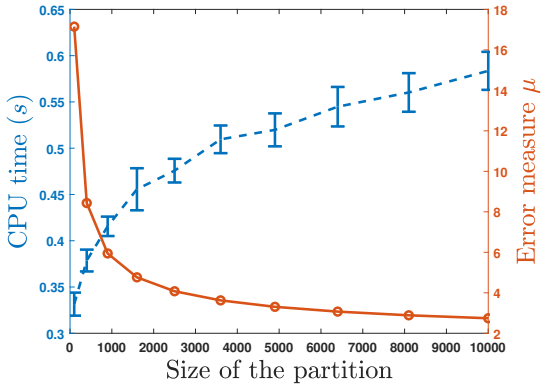

Fig. 10. Line in blue represents the average time of execution with respect to the number of cells in the partitions, with bars representing the standard deviation. Line in orange represents the performance criterion $\boldsymbol{\mu(\mathcal{D}, Q)}$ with respect to the number of cells in the partitions.

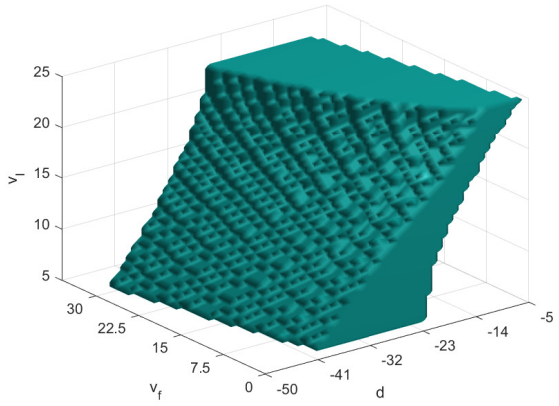

Fig. 11. Controllable set of the symbolic controller enforcing the assume-guarantee contract (39) for system (38)

using different randomly sampled data sets. Figure 10 shows the results of the experiment. Time of execution increases logarithmically when $|Q|$ the number of the partition elements is small. Then, the relation becomes linear for big values of $|Q|$. This behavior can be justified by the complexity relation introduced in Proposition 6. Similar to the first experiment, the performance criterion decreases with the increase in the number of partition elements.

3) Comparison with the state-of-the-art robust models: We have already shown both theoretically in Proposition 6 and experimentally the computational complexity of calculating the introduced data-driven model. Let us compare with the set membership approaches in [27] and [4], where an optimization problem is used to find the model. The complexity of this approach grows polynomially with the number of data points. In contrast, our approach scales linearly. The maximum reported number of data points used in [27] is in the tens of thousands. The number of points used is less in [4]. As can be seen, the introduced approach in this paper can handle orders of magnitude more data points. The same can be said about [30], where a Mixed Integer Linear Programming (MILP) problem is used to find the model, and the reported number of data points used in the case study is 400.

B. Cruise control problem

The second example showcases a system that can be seen as a sum of a known function and an unknown monotone function. We find the abstraction representing the system. Then, we use this abstraction to synthesize a safety controller.

Let us consider two vehicles moving in one lane on an infinite straight road. The leader is uncontrollable (vehicle 2), whereas the follower is controllable (vehicle 1). A discrete-time model of this setup is given by equations:

$$\begin{aligned} d(k+1) &= d(k) + (v_f(k) - v_l(l))T_0 \\ v_f(k+1) &= v_f(k) + \gamma(u(k), v_f(k))T_0 \\ v_l(k+1) &= v_l(k) + w(k)T_0 \end{aligned} \tag{38}$$

Here u is the control input, and d is the signed distance between the vehicles. v_l, v_f are the velocities of the leader and follower, respectively. The function γ represents the follower vehicle acceleration caused by the control input and the friction forces acting upon it. The term $w(k) \in [\underline{W}, \overline{W}]$ accounts for uncertainty in the leader velocity and is considered as a disturbance. Model (38) can be seen as the sum of a known part, which can be inferred from the physics of the system, and an unknown or hard-to-model part, namely the function $\gamma(u(k), v_f(k))T_0$. We are also able to study the monotonicity of the function $\gamma(u(k), v_f(k))T_0$ starting from the physics of the system. The acceleration of the car will increase with the increase of the input, and the friction forces will increase with the increase of the velocity. Therefore, by making the change of variable $v'_f = -v_f$ we can apply the algorithm introduced in this work to find an abstraction of the system.

Function γ is given as follows

$$\gamma(u, v) = \; u - M_C^{-1}(f_0 + f_1 v + f_2 v^2).$$

The vector of parameters $f = (f_0, f_1, f_2) \in \mathbb{R}^3_+$ describes road friction and vehicle aerodynamics whose numerical values are taken from [26]: $f_0 = 51\ N$, $f_1 = 1.2567\ Ns/m$, $f_2 = 0.4342 Ns^2/m^2$. For the car mass, we chose $M_C = 1370\ kg$. Other numerical values related to the model are $X = [-50, -5] \times [0, 30] \times [5, 25]$, $U = [-3, 3], T_0 = 0.7\ s$, $\underline{W} = -\overline{W} = 2\ m/s^2$. To build the abstraction we fixed a partition $K_x^1 = 50$, $K_x^2 = 50$, $K_x^3 = 50$, $K_u^1 = 50$, $c = 0.005$, and sampled a set containing $|\mathbb{K}| = 10^6$ data points. Finding the over-approximation of the unknown monotone part was done in $0.633s$.

To show the usefulness of the calculated abstraction, we use it to synthesize a safety controller. In this example, the leader's vehicle is uncontrollable. Therefore, we considered a specification given as an assume-guarantee contract:

$$\begin{aligned} \forall k \in \mathbb{N}\ v_l(k) \in [\underline{v}_l, \overline{v}_l] \implies \\ \forall k \in \mathbb{N},\ v_f(k) \in [\underline{v}_f, \overline{v}_f] \wedge d(k) \in [\underline{d}, \overline{d}]. \end{aligned} \tag{39}$$

If the velocity of the leader remains within the bounds $[\underline{v}_l, \overline{v}_l]$, then the velocity of the follower and the distance between the two vehicles should remain within the bounds $[\underline{v}_f, \overline{v}_f]$ and $[\underline{d}, \overline{d}]$, respectively. The synthesis of symbolic controllers enforcing assume-guarantee contracts such as (39) has been

considered in [31]. For this example, we chose $\underline{d} = \alpha_1^1$, $\overline{d} = \alpha_{K^1}^1$, $\underline{v}_f = \alpha_1^2$, $\overline{v}_f = \alpha_{K^2}^2$, $\underline{v}_l = \alpha_1^3$, $\overline{v}_f = \alpha_{K^3}^3$. The controllable set of the resulting symbolic controller is shown in Figure 11.

C. Lorenz system

In the last example, we study the Lorenz system, a system with bounded derivatives. We first compute the bounds on the disturbance and on the derivatives of the function representing the system from the data. Then, we find a data-driven abstraction that we use to synthesize a controller for the system. The controlled Lorenz system has three states and one input. We built the abstraction using a large number of points and a partition with a large number of elements, and we showed that we could do it efficiently.

The discrete-time controlled Lorenz system is described by the following model

$$\begin{aligned} x(k+1) &= x(k) + \sigma(x(k) - y(k) + w_1(k))T_0 \\ y(k+1) &= y(k) + (\rho\ x(k) - x(k)z(k) - \\ &\qquad y(k) + u(k) + w_2(k))T_0 \\ z(k+1) &= z(k) + (x(k)y(k) - \beta\ z(k) + w_3(k))T_0 \end{aligned} \tag{40}$$

For the particular parameter values of $\sigma = 10, \beta = \frac{8}{3}, \rho = 28$ the system behaves chaotically [19].

We studied the system on the sets $X = [-10, 10] \times [-10, 10] \times [-10, 10]$, $U = [-200, 200]$, $\overline{\mathbf{w}} = [0.5, 0.5, 0.5]$. The discretization time was chosen to be $T_0 = 0.01s$. First, we used the algorithm described in Section IV to calculate the bounds on the disturbance and on the derivatives. In the first step, we used $5 * 10^4$ data points sampled randomly. In the second step, we sampled 10^6 data points. The results of the calculated bounds on Jacobian matrices and the disturbance are shown below. Subscript $\mathcal{D}$ refers to matrices calculated from data, while their unsubscripted counterparts are the actual matrices we are estimating. The results are rounded to the third significant digit.

$$\overline{J}^x = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.38 & 0.99 & 0.1 \\ 0.1 & 0.1 & 0.973 \end{bmatrix}, \overline{J}^x_{\mathcal{D}} = \begin{bmatrix} 0.9 & 0.1 & 1.64{*}10^{-6} \\ 0.377 & 0.99 & 9.46{*}10^{-2} \\ 9.09{*}10^{-2} & 9.31{*}10^{-2} & 0.975 \end{bmatrix}$$

$$\underline{J}^x = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.18 & 0.99 & -0.1 \\ -0.1 & -0.1 & 0.973 \end{bmatrix},$$

$$\underline{J}^x_{\mathcal{D}} = \begin{bmatrix} 0.9 & 0.1 & -2.1{*}10^{-7} \\ 0.185 & 0.989 & -9.27{*}10^{-2} \\ -9.3{*}10^{-2} & -9.2{*}10^{-2} & 0.97 \end{bmatrix}$$

$$\overline{J}^u = \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix}, \overline{J}^u_{\mathcal{D}} = \begin{bmatrix} -1.9{*}10^{-7} \\ 9.95{*}10^{-3} \\ 2.88{*}10^{-6} \end{bmatrix}$$

$$\underline{J}^u = \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix}, \underline{J}^u_{\mathcal{D}} = \begin{bmatrix} -1.9{*}10^{-7} \\ 9.95{*}10^{-3} \\ 4.58{*}10^{-5} \end{bmatrix}$$

$$\overline{\mathbf{w}}{*}T_0 = \begin{bmatrix} 0.005 \\ 0.005 \\ 0.005 \end{bmatrix}, \overline{\mathbf{w}}_{\mathcal{D}} * T_0 = \begin{bmatrix} 0.005 \\ 5.85{*}10^{-2} \\ 5.76{*}10^{-2} \end{bmatrix}$$

The execution time of the first step is $12.4s$, whereas the execution time for the second step is $0.5s$. From the results, we can see that all values but the disturbances added to the second and third states are similar to the actual values. The second and third values of the disturbance are more conservative. If we choose $\beta = 10^{-6}$, then according to Proposition 9, with a probability at least $1 - \beta$, it holds that the probability of finding two points violating the calculated bounds is less than $2 * 10^{-5}$.

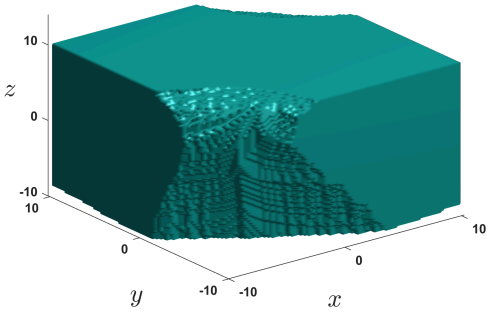

Fig. 12. The maximal controlled invariant of (40) calculated using the data-driven abstraction

We used these values to build the data-driven abstraction. We sampled 10^8 data points. We choose the parameters of the partition as follows $K_x^1 = 100$, $K_x^2 = 100$, $K_x^3 = 100$, $K_u^1 = 100$, $c = 0.05$. The execution time to reach the abstraction was $929.2\ s$. To test the validity of the calculated abstraction, we used it to find the maximal safe controlled invariant of the system. Given a safe set $X_s \subseteq X$, a safe controlled invariant $\mathcal{I}_s$ is a set included in the safe set $\mathcal{I}_s \subseteq X_s$, and that can be rendered invariant using a suitable controller. Using the finite-state abstraction, we calculated earlier, we can apply an iterative algorithm to find the maximal safe controlled invariant [33]. We chose the safe set to be $X_s = X$. The resulting maximal control invariant is represented in Figure 12.

VI. Conclusion

In this work, data-driven over-approximation of monotone maps has been introduced. We have proved that the introduced over-approximating maps are minimal. We have also studied the case of over-approximating a monotone map on a fixed partition. We have developed an efficient algorithm to find those maps and then used this data-driven over-approximation to build models for partially unknown systems, where the unknown part is monotone. Using the data-driven models, symbolic abstractions of the system are calculated, which then can be used to synthesize discrete correct-by-design controllers. We have extended our method to study systems with bounded derivatives and introduced a procedure to calculate those bounds from data. In the numerical examples part, we have shown that the time to calculate the over-approximation grows linearly with the number of data points and sub-linearly with the number of elements in the partition. We have also shown the validity of the calculated abstraction by synthesizing safety controllers. In the future, we want to refine and improve the over-approximation using data collected online. We also want to safely explore unknown parts of the state space from which we do not have data points.

References

[1] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*. Springer, 2017.

[2] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[3] M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[4] M. Canale, L. Fagiano, and M. Signorile. Nonlinear model predictive control from data: a set membership approach. *International Journal of Robust and Nonlinear Control*, 24(1):123–139, 2014.

[5] R. Coppola, A. Peruffo, and M. Mazo Jr. Data-driven abstractions for verification of deterministic systems. *arXiv preprint arXiv:2211.01793*, 2022.

[6] A. Devonport, A. Saoud, and M. Arcak. Symbolic abstractions from data: a PAC learning approach. In *IEEE Conference on Decision and Control*, pages 599–604, 2021.

[7] A. Girard. Controller synthesis for safety and reachability via approximate bisimulation. *Automatica*, 48(5):947–953, 2012.

[8] A. Girard and A. Eqtami. Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. *Automatica*, 127, 2021.

[9] A. Girard, G. Pola, and P. Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2009.

[10] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. Dimarogonas. Learning-based symbolic abstractions for nonlinear control systems. *arXiv preprint arXiv:2004.01879*, 2020.

[11] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

[12] Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[13] E. Ivanova, A. Saoud, and A. Girard. Lazy controller synthesis for monotone transition systems and directed safety specifications. *Automatica*, 135:109993, 2022.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis, 2001*. Springer, 2001.

[15] Y. Kawano, B. Besselink, J. M. A. Scherpen, and M. Cao. Data-driven model reduction of monotone systems by nonlinear DC gains. *IEEE Transactions on Automatic Control*, 65(5):2094–2106, 2020.

[16] M. Kazemi, R. Majumdar, M. Salamati, S. Soudjani, and B. Wooding. Data-driven abstraction-based control synthesis. *arXiv preprint arXiv:2206.08069*, 2022.

[17] M. Khajenejad, Z. Jin, and S. Z. Yong. Interval observers for simultaneous state and model estimation of partially known nonlinear systems. In *American Control Conference*, pages 2848–2854, 2021.

[18] A. Lavaei and E. Frazzoli. Data-driven synthesis of symbolic abstractions with guaranteed confidence. *IEEE Control Systems Letters*, 7:253–258, 2022.

[19] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[20] A. Makdesi, A. Girard, and L. Fribourg. Data-driven abstraction of monotone systems. In *Learning for Dynamics and Control*, pages 803–814. PMLR, 2021.

[21] A. Makdesi, A. Girard, and L. Fribourg. Efficient data-driven abstraction of monotone systems with disturbances. *IFAC-PapersOnLine*, 54(5):49–54, 2021.

[22] A. Makdesi, A. Girard, and L. Fribourg. Safe learning-based model predictive control using the compatible models approach. *European Journal of Control*, page 100849, 2023.

[23] P.-J. Meyer, A. Devonport, and M. Arcak. *Interval reachability analysis: Bounding trajectories of uncertain systems with boxes for control and verification*. Springer Nature, 2021.

[24] P.-J. Meyer, A. Girard, and E. Witrant. Safety control with performance guarantees of cooperative systems using compositional abstractions. *IFAC-PapersOnLine*, 48(27):317–322, 2015.

[25] M. Milani Fard, K. Canini, A. Cotter, J. Pfeifer, and M. Gupta. Fast and flexible monotonic functions with ensembles of lattices. *Advances in neural information processing systems*, 29, 2016.

[26] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2015.

[27] C. Novara and M. Milanese. Set membership identification of nonlinear systems. In *IEEE Conference on Decision and Control*, volume 3, pages 2831–2836, 2000.

[28] G. Pola and M. D. Di Benedetto. Control of cyber-physical-systems with logic specifications: a formal methods approach. *Annual Reviews in Control*, 47:178–192, 2019.

[29] G. Pola, T. Masciulli, E. De Santis, and M. D. Di Benedetto. Data-driven controller synthesis for abstract systems with regular language specifications. *Automatica*, 134:109903, 2021.

[30] S. Sadraddini and C. Belta. Formal guarantees in data-driven model identification and control synthesis. In *International Conference on Hybrid Systems: Computation and Control*, pages 147–156, 2018.

[31] A. Saoud, A. Girard, and L. Fribourg. Contract-based design of symbolic controllers for safety in distributed multiperiodic sampled-data systems. *IEEE Transactions on Automatic Control*, 66(3):1055–1070, 2020.

[32] V. Sinyakov and A. Girard. Formal controller synthesis from specifications given by discrete-time hybrid automata. *Automatica*, 131, 2021.

[33] P. Tabuada. *Verification and control of hybrid systems: A symbolic approach*. Springer Science & Business Media, 2009.

[34] L. Yang, O. Mickelin, and N. Ozay. On sufficient conditions for mixed monotonicity. *IEEE Transactions on Automatic Control*, 64(12):5080–5085, 2019.

[35] M. Zamani, G. Pola, M. Mazo, and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2011.

[36] D. Zonetti, A. Saoud, A. Girard, and L. Fribourg. Decentralized monotonicity-based voltage control of dc microgrids with zip loads. *IFAC-PapersOnLine*, 52(20):139–144, 2019.

Anas Makdesi received the B.Sc. degree in Electronic Systems Engineering from the Higher Institute for Applied Sciences and Technology (HIAST), Damascus, Syria, in 2017, and the M.Sc. degree in Control, Signal, and Image Processing in 2020 from Paris Saclay University, Paris, France, where he is currently working toward the Ph.D. degree in Control System engineering. His research interests include data-driven approaches for cyber-physical systems and their applications in robotics.

Antoine Girard is a Senior Researcher at CNRS and a member of the Laboratory of Signals and Systems. He received the Ph.D. degree in applied mathematics from Grenoble Institute of Technology, in 2004. From 2004 to 2006, he held postdoctoral positions at University of Pennsylvania and Université Grenoble-Alpes. From 2006 to 2015, he was an Assistant/Associate Professor at the Université Grenoble-Alpes. His main research interests deal with analysis and control of hybrid systems with an emphasis on computational approaches, formal methods and applications to cyber-physical systems. Antoine Girard received the George S. Axelby Outstanding Paper Award from the IEEE Control Systems Society in 2009. In 2014, he was awarded the CNRS Bronze Medal. In 2015, he was appointed as a junior member of the Institut Universitaire de France (IUF). In 2016, he was awarded an ERC Consolidator Grant. In 2018, he received the first HSCC Test of Time Award and the European Control Award.

Laurent Fribourg received a civil engineering degree in 1980 from Aeronautics School Sup'Aero (Toulouse), and a PhD in computer science in 1982 from University Paris 7. He was appointed by CNRS in 1984 as a full-time researcher. He co-founded the CNRS laboratory of computer science LSV at ENS Cachan (now ENS Paris-Saclay) in 1997, and became head of the CNRS interdisciplinary Institut Farman at ENS Paris-Saclay in 2018. He has written more than 100 papers in the area of formal methods, several of them in collaboration with industrial partners.