Data driven finite abstractions by simulation relations with probabilistic guarantees using regression trees

Data driven finite abstractions by simulation relations with probabilistic guarantees using regression trees

Alessandro D'Innocenzo[1] Khalil Ul Rehman[1,2] Yuriy Zacchia Lun[1] ,

***Abstract*— In this paper we propose a novel methodology to construct, given trajectories measured from a dynamical system, a finite abstraction by means of a transition system. We prove that our abstraction is a *simulation* of the original dynamical system, providing quantified probabilistic guarantees derived using the scenario approach. We test our methodology on a benchmark on hybrid systems showing that it strongly reduces the cardinality of the abstraction states with respect to a uniform grid, and is thus very promising for handling abstractions of large dimensional systems.**

I. Introduction

Verification of cyber-physical systems received an increasing attention in the last years [1], [2]. In this context, Model checking is a method for verifying some properties on the executions of a given system by means of temporal logic formulae. It has been applied with great impact, for example, in verifying the safety [3] and correctness [4] of a system. In order to apply Model checking techniques to a black-box dynamical system starting from data measurements it is necessary to identify the systems dynamics and construct a finite abstraction, possibly with certified probabilistic guarantees. In what follows, we provide a survey of related literature and illustrate the novelty of our work with respect to the state of the art.

State-of-the-Art. The literature on construction techniques of finite abstractions of dynamical systems is very rich, and we try to provide a summary of the main related works. Simulation relation is a mathematical tool to relate the behavior of a dynamical system and a finite abstraction. It helps to evaluate how closely a simplified abstraction can replicate the behavior of a complex system, and is very helpful in verification of relevant properties when exact equivalence of the abstraction results computationally intractable: for this reason, the scientific community investigates novel methodologies and frameworks to construct finite abstractions of dynamical systems with reasonable complexity. These notions have been largely investigated for verification and control of hybrid dynamical systems, see [5] and references therein. In [6] a comprehensive framework that characterizes various simulation relations in the context of abstraction-based control design has been proposed, based on the concept of an augmented system that enables a feedback refinement relation of the concrete system with the abstract system. Recently, on the wave of the recent advancements of Machine Learning and its exploitation in control applications, several methods have been proposed on the construction of data driven abstractions. In [7] a method is proposed for constructing abstractions of dynamical systems using data-driven approaches, providing probably approximately correct (PAC) guarantees for system behavior inclusion. In [8] a framework has been proposed to construct a finite abstraction of an unknown discrete-time deterministic control system, where alternating bisimulation functions (ABF) have been constructed as a robust optimization problem, which is then approximated by a scenario optimization program based on the trajectory dataset. In [9] a sample-based, sequential method to abstract a potentially black-box dynamical system using a sequence of memory-dependent Markov chains of increasing size is proposed. In [10] a data-driven approach for abstracting monotone dynamical systems is introduced, facilitating the use of symbolic control techniques for controller synthesis. In [11] a data-driven approach to computing finite bisimulations for state transition systems with very large, possibly infinite state spaces is proposed based on stutter-insensitive bisimulations of deterministic systems and ranking functions from sample states. In [12] a data-driven approach for constructing finite abstractions of continuous-time control systems with unknown dynamics is proposed, facilitating controller synthesis with formal correctness guarantees. In [13] a framework that utilizes input-output data from unknown systems to synthesize controllers based on signal temporal logic (STL) specifications is developed, ensuring formal guarantees in the control synthesis process. Previous works related to this paper are [14], where system identification techniques are combined with machine learning to derive a predictive model of a dynamical system, and [15], where probabilistic guarantees are formulated on the accuracy of such model.

Paper contribution. The *leit-motiv* in the research on constructing finite abstractions of dynamical systems is reducing the cardinality of the abstraction, as uniform grids of the state space are impractical as the dimension of the state variable increases. The main contributions of this paper are the following:

1) we provide a novel algorithm, that takes as input the predictive model derived using [15, Algorithm 2] from

This work was partially supported by: the Italian government through CIPE Resolution 70/2017 (Centre EX-Emerge); the Italian National Recovery and Resilience Plan of NextGenerationEU through the MoVeOver/SCHEDULE project (CUP J33C22002880001); the EU through projects DigInTraCE (GA 101091801) and Resilient Trust (GA 101112282).

[1] Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 AQ, Italy. alessandro.dinnocenzo@univaq.it, yuriy.zacchialun@univaq.it, khalilul.rehman@student.univaq.it

[2] Politecnico di Torino, 10129 Torino, Italy, khalil.rehman@polito.it

a set of trajectories measured from a dynamical system, to construct a finite abstraction by means of a transition system. The main intuition behind our approach is that the CART algorithm generates a *smart* partition of the state space via hyper-rectangles optimizing the prediction accuracy of the next state in each equivalence class, thus generating a non-uniform grid: to the best of the authors' knowledge, this approach is a novel contribution to the scientific literature, alternative to the existing methodologies;

2) we prove that our abstraction is a *simulation* of the original dynamical system and provide quantified probabilistic guarantees derived using the well known *scenario approach* (see [16] and references therein). More precisely, we provide a bound to the probability that some trajectories of the real system (i.e. trajectories not belonging to the training dataset) do not satisfy the simulation relation obtained on the training dataset;
3) we validate our methodology on a benchmark on hybrid systems [17] showing that, as we expected, the hyper-rectangular partition constructed using the CART algorithm adapts to the system model, generating a partition that strongly decreases the cardinality of equivalence classes with respect to a uniform grid. This property is very promising in the perspective of providing a new solution for the enduring problem of complexity explosion when constructing finite abstractions of large dimensional systems, and we believe that the methodology proposed in this paper provides a new approach to tackle the complexity issue.

The paper is organised as follows: in Section II we introduce the mathematical background; in Section III we illustrate our proposed methodology and prove the properties of the obtained finite abstraction; in Section IV we validate the effectiveness of our algorithm on a benchmark on hybrid dynamical systems.

II. Mathematical background

In this section we introduce the transition system modelling framework, the definition of simulation and bisimulation relations, the scenario approach and the CART algorithm.

A. Transition systems, (bi)simulation relation.

We first introduce transition systems, which enables to model discrete, continuous and hybrid systems [18].

Definition 1: A (labeled) transition system with observations is a tuple $\mathcal{S} = (Q, \Sigma, \rightarrow, Q^0)$ that consists of: a (possibly infinite) set Q of states; a (possibly infinite) set Σ of labels; a transition relation $\rightarrow \subseteq Q \times \Sigma \times Q$; a (possibly infinite) set $Q^0 \subseteq Q$ of initial states.
Let $\mathcal{S}_1 = (Q_1, \Sigma, \rightarrow_1, Q_1^0)$ and $\mathcal{S}_2 = (Q_2, \Sigma, \rightarrow_2, Q_2^0)$ be two labeled transition systems with the same set of labels.

Definition 2: A relation $\Gamma \subseteq Q_1 \times Q_2$ is called a simulation relation of $\mathcal{S}_1$ by $\mathcal{S}_2$ if, for all $(q_1, q_2) \in \Gamma$ and for all $q_1 \xrightarrow{\sigma} q_1'$, there exists $q_2 \xrightarrow{\sigma} q_2'$ such that $(q_1', q_2') \in \Gamma$.

Such a relation is called a bisimulation when a relation is both a simulation of $\mathcal{S}_1$ by $\mathcal{S}_2$ and a simulation of $\mathcal{S}_2$ by $\mathcal{S}_1$.

Definition 3: $\mathcal{S}_2$ simulates $\mathcal{S}_1$ (denoted $\mathcal{S}_1 \preceq \mathcal{S}_2$) if there exists Γ, a simulation relation of $\mathcal{S}_1$ by $\mathcal{S}_2$, such that for all $q_1 \in Q_1^0$, there exists $q_2 \in Q_2^0$ such that $(q_1, q_2) \in \Gamma$.
If any initial state of $\mathcal{S}_1$ can be related to any initial state of $\mathcal{S}_2$ and conversely, then $\mathcal{S}_1$ and $\mathcal{S}_2$ simulate each other. We say that $\mathcal{S}_1$ and $\mathcal{S}_2$ are bisimilar.

Definition 4: Given an autonomous nonlinear dynamical system $\mathcal{D}$ defined by

$$x(k+1) = f(x(k)), x \in \mathbb{R}^m, x(0) \in X^0, \tag{1}$$

we define the corresponding equivalent transition system model $\mathcal{S}_\mathcal{D} = (Q_\mathcal{D}, \Sigma_\mathcal{D}, \rightarrow_\mathcal{D}, Q_\mathcal{D}^0)$, with $Q_\mathcal{D} = \mathbb{R}^m$, $\Sigma_\mathcal{D} = \varnothing$, $x \rightarrow_\mathcal{D} x' \Leftrightarrow x' = f(x)$, $Q_\mathcal{D}^0 = X^0$.

B. Scenario approach

We now provide the mathematical background on the scenario approach, which we will exploit to derive probabilistic guarantees on the technique proposed in this paper. For more details the reader is referred to [16] and references therein.

Let us consider a probability space $(\Delta, \mathcal{D}, \mathcal{P})$ and a sample $(\delta^{(1)}, \ldots, \delta^{(N)}) \in \Delta^N$ of N elements drawn independently from Δ according to the same probability measure $\mathcal{P}$. We call each observation $\delta^{(\cdot)}$ a *scenario*. Let us also consider a set $\Theta \subseteq \mathbb{R}^d$, called decision space, and define a function $\mathcal{A}_N : \Delta^N \rightarrow \Theta$ denoted as the *scenario decision* $\theta_N^* := \mathcal{A}_N(\delta^{(1)}, \ldots, \delta^{(N)})$. Let the scenario decision $\mathcal{A}_N$ be a unique solution θ_N^*, possibly after applying a tie-break rule, of a constrained optimization program:

$$\begin{aligned} &\min_{\theta \in \Theta} f(\theta) \\ &\text{subject to } \theta \in \Theta_{\delta^{(i)}}, i = 1, \ldots, N, \end{aligned} \tag{2}$$

where f, Θ and $\Theta_{\delta^{(\cdot)}}$ can be any function and constraint (i.e., convex or nonconvex). Let us define the *violation probability* of a decision $\theta \in \Theta$ as

$$V(\theta) = \mathcal{P}\{\delta \in \Delta : \theta \notin \Theta_\delta\}. \tag{3}$$

For a given reliability parameter $\epsilon \in (0, 1)$, we say that $\theta \in \Theta$ is ϵ-feasible if $V(\theta) \leq \epsilon$. The violation of a scenario decision $V(\theta_N^*)$ is a random variable over Δ^N. The idea behind the scenario approach framework is to characterise the distribution of $V(\theta_N^*)$ and find a confidence bound $1 - \beta$ such that the relation $V(\theta) \leq \epsilon$ is satisfied.

1) Convex case: When the optimization program (2) is convex, as the deepest result it has been shown in [19] that $V(\theta_N^*)$ is dominated by a Beta distribution, i.e.

$$\mathcal{P}^N\{V(\theta_N^*) > \epsilon\} \leq \beta, \ \beta = \sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}, \tag{4}$$

where we recall that d is the dimension of the optimization variable, and that the only assumption on the probability space is that each scenario is drawn independently.

2) Nonconvex case: When the optimization program (2) is nonconvex, a different approach is used in [16]. Assume that $\mathcal{A}_N(\delta^{(1)}, \dots, \delta^{(N)})$ is the (unique, possibly suboptimal) scenario decision given a sample $(\delta^{(1)}, \dots, \delta^{(N)})$: we define a support subsample $(\delta^{(i_1)}, \dots, \delta^{(i_k)}), k \leq N$, as a k-tuple of elements of $(\delta^{(1)}, \dots, \delta^{(N)})$ such that $\mathcal{A}_k(\delta^{(i_1)}, \dots, \delta^{(i_k)}) = \mathcal{A}_N(\delta^{(1)}, \dots, \delta^{(N)})$. Let a support subsample of cardinality k exist, then the following holds:

$$\mathcal{P}^N\{V(\theta_N^*) > \epsilon(k)\} \leq \beta$$
$$\epsilon(k) = \begin{cases} 1 & \text{k=N,} \\ 1 - \left(\frac{\beta}{N\binom{N}{k}}\right)^{(N-k)^{-1}} & \text{otherwise.} \end{cases} \quad (5)$$

C. CART algorithm

We briefly recall the partitioning algorithm of CART, and refer the reader to [20] for more details. In a supervised learning framework, we consider an input dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ and an output dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ of N samples each, where $y_i \in \mathbb{R}^n$ and $x_i \in \mathbb{R}^m$. The final goal of CART is to identify a function $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^n$ to estimate $\hat{y} = \mathcal{T}(x)$: to this aim, the dataset is iteratively partitioned, according to a tree graph structure grown during the algorithm, by a set of hyper-rectangles $R_1, \dots, R_{|\mathcal{T}|}$ that induce a partition of the input space $\mathbb{R}^m$: to each hyper-rectangle corresponds one (and only one) leaf of the grown tree graph.

More in detail, without any loss of generality we restrict our attention to recursive binary partition of the CART algorithm [20] extended to handle multivariate outputs (see e.g. [21], [22]). Starting with the whole dataset, which is associated to the root node of the tree, consider a split variable j over the n available and a split point s, and define the half-spaces $R_L(j,s) = \{x_i \mid x_{i,j} < s\}$ and $R_R(j,s) = \{x_i \mid x_{i,j} \geq s\}$, where $x_{i,j}$ is the j-th component of sample $x_i \in \mathbb{R}^m$. The CART algorithm solves the following optimization problem to find the optimal ι^* and s^*:

$$\min_{\iota,s}\left[\min_{c_L}\sum_{x_i\in R_L(\iota,s)}||y_i - c_L||^2 + \min_{c_R}\sum_{x_i\in R_R(\iota,s)}||y_i - c_R||^2\right]. \quad (6)$$

Once the best split is found the dataset is partitioned into the two resulting regions, and the splitting procedure is repeated starting from each of the nodes associated to the new defined regions. The process is repeated until a stopping criterion is applied, e.g. the tree size is a tuning parameter that should be chosen to avoid overfitting and variance phenomena: in this paper we will use, as stopping criterion, the minimum number of samples contained in each leaf. Then, $\hat{y}$ is estimated in each leaf ℓ_j using a constant c_{ℓ_j} given by the average of the samples in the partition.

In the rest of this work we will denote with $\mathcal{T}$ the regression tree, with ℓ_j the j^{th} leaf of $\mathcal{T}$, with $|\mathcal{T}|$ the number of leaves of $\mathcal{T}$ and with $|\ell_j|$ the number of samples in the leaf ℓ_j. Also, we will denote with $\hat{y} = \mathcal{T}(x)$ the regression tree prediction to a new sample x given by

$$\mathcal{T}(x) = \sum_{j=1}^{|\mathcal{T}|} c_j I\{x \in R_j\}, \quad (7)$$

with $c_j = \text{ave}(y_i | x_i \in \ell_j)$ the prediction associated to leaf ℓ_j, with R_j the hyper-rectangular partition set associated to leaf ℓ_j and with $I\{x \in R_j\}$ the indicator function that is equal to 1 if $x \in R_j$, and 0 otherwise.

III. Finite abstraction via simulation relation with probabilistic guarantees

In this section we first recall for completeness Algorithm 0[1] as in [15, Algorithm 2] that constructs, given a dataset of trajectories generated by a dynamical system, a piecewise affine AR predictive model based on CART with probabilistic guarantees. Then we propose a novel technique, Algorithm 1, to construct, given the output of Algorithm 0, a finite transition system, and prove that the constructed abstraction is a simulation of the original system with quantified probabilistic guarantees.

Consider a dynamical system $\mathcal{D}$ as in Equation (1) representing a nonlinear AR model, i.e. with $x(k) = [z'(k-1), \dots, z'(k-r)]'$, $x \in \mathbb{R}^m, z \in \mathbb{R}^n, m = rn$, and with unknown dynamics f. Consider a set of N sample trajectories $\{z_i(0), \dots, z_i(r)\}_{i=1}^N$ extracted from $\mathcal{D}$. All trajectories, each considered as a scenario $\delta^{(i)}$, are assumed to be collected as independent drawns from a same probability measure in order to satisfy the conditions required for applying the scenario approach bounds. Let us define an input (features) dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ with $x_i := (z_i(0), \dots, z_i(r-1)) \in \mathbb{R}^m, m = rn$, and an output (response) dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ with $y_i := z_i(r) \in \mathbb{R}^n$.

Algorithm 0 takes as input the datasets $\mathcal{X} = \{x_i\}_{i=1}^N, \mathcal{Y} = \{y_i\}_{i=1}^N$, and is based on the idea that the CART algorithm is applied once on the whole dataset (line 3), and in each leaf ℓ_j the optimization problem (9) is solved to determine an affine predictive model $\alpha_j^*, \alpha_{j,0}^*$ (instead of a constant value c_{ℓ_j}). Then, candidate support subsample sets are searched within each leaf by removing, one by one, samples in the leaf (lines 7-8) until the reduced set satisfies the support subsample property (lines 9-13). The output is a regression tree $\mathcal{T}$, with leaves $\ell_1, \dots, \ell_{|\mathcal{T}|}$ each associated to a hyper-rectangle $R_1, \dots, R_{|\mathcal{T}|}$ which provides a partition of the state space. An affine predictive model is associated to each leaf by means of $\alpha_j^* \in \mathbb{R}^{n\times m}, \alpha_{j,0}^* \in \mathbb{R}^n$, therefore the predictive function associated to $\mathcal{T}$ is given by

$$\mathcal{T}(x) = \sum_{j=1}^{|\mathcal{T}|}(\alpha_j^* x + \alpha_{j,0}^*) \cdot I\{x \in R_j\}. \quad (8)$$

Beyond the tree structure $\mathcal{T}$ and $\{\alpha_j^*, \alpha_{j,0}^*\}_{j=1}^{|\mathcal{T}|}$, Algorithm 0 also provides the variables $\{h_j^*, k_j^*, |\ell_j|\}_{j=1}^{|\mathcal{T}|}$, which respectively consist of, for each leaf j, the worst-case error prediction error h_j^*, the cardinality of the largest found support subsample k_j^*, and the cardinality of samples $|\ell_j|$ contained in leaf j. In Algorithm 0 we denote, for the j-th leaf, by $\ell_j = \{x_{j_1}, \dots, x_{j_{|\ell_j|}}\}$ the input samples in ℓ_j and by $\{y_{j_1}, \dots, y_{j_{|\ell_j|}}\}$ the corresponding output samples.

[1]Algorithm 0 slightly differs from [15, Algorithm 2] since the left-hand side of constraint (9) consists of the norm of a vector, i.e. Euclidean or infinity norm, instead of a scalar

Algorithm 0 [15] Compute predictive model $\mathcal{T}$ given $\mathcal{D}$

1: **Input:** Datasets $\mathcal{X} = \{x_i\}_{i=1}^N, \mathcal{Y} = \{y_i\}_{i=1}^N$
2: **Output:** $\mathcal{T}, \{\alpha_j^*, \alpha_{j,0}^*, h_j^*, k_j^*, |\ell_j|\}_{j=1}^{|\mathcal{T}|}$
3: Apply CART algorithm to $(\mathcal{X}, \mathcal{Y})$, obtaining as output a tree structure $\mathcal{T}$ with leaves $\{\ell_j\}_{j=1}^{|\mathcal{T}|}$
4: **for all** $j = 1, \dots, |\mathcal{T}|$ **do**
5: Compute in leaf ℓ_j an error bound h_j^* and an affine predictor $\alpha_j^* \in \mathbb{R}^{n\times m}, \alpha_{j,0}^* \in \mathbb{R}^n$ by solving the QP:

$$\min_{h_j, \alpha_j, \alpha_{j,0}} h_j$$
$$\text{subject to } ||y_{j_i} - \alpha_j x_{j_i} - \alpha_{j,0}||^2 \le h_j, \forall x_{j_i} \in \ell_j \quad (9)$$

6: Set $X_j := \ell_j, Y_j := \{y_{j_i}\}_{i=1}^{|\ell_j|}, k_j^* := |X_j|$
7: **while** $X_j \neq \varnothing$ **do**
8: **for all** $\zeta = 1, \dots, |X_j|$ **do**
9: Compute in leaf ℓ_j an error bound $\tilde{h}_j^*$ and an affine predictor $\tilde{\alpha}_j^* \in \mathbb{R}^{n\times m}, \tilde{\alpha}_{j,0}^* \in \mathbb{R}^n$ by solving:

$$\min_{\tilde{h}_j, \tilde{\alpha}_j, \tilde{\alpha}_{j,0}} \tilde{h}_j$$
$$\text{subject to } ||y_{j_i} - \tilde{\alpha}_j x_{j_i} - \tilde{\alpha}_{j,0}||^2 \le \tilde{h}_j, \forall x_{j_i} \in X_j \setminus x_\zeta \quad (10)$$

10: **if** $h_j^* = \tilde{h}_j^*, \alpha_j^* = \tilde{\alpha}_j^*, \alpha_{j,0}^* = \tilde{\alpha}_{j,0}^*$ **then**
11: Set $X_j := X_j \setminus \{x_\zeta\}, Y_j := Y_j \setminus \{y_\zeta\}$
12: Set $k_j^* := |X_j|$
13: Exit **For** cycle *(support subsample found)*
14: **else** if $\zeta = |X_j|$ Exit **For** and **While** cycles *(no support subsample found)*
15: **end if**
16: **end for**
17: **end while**
18: **end for**
19: **Return** $\mathcal{T}, \{\alpha_j^*, \alpha_{j,0}^*, h_j^*, k_j^*, |\ell_j|\}_{j=1}^{|\mathcal{T}|}$

As the main result of this paper we first propose Algorithm 1, to construct a finite transition system abstraction $\mathcal{S}_\mathcal{T}$ of a system $\mathcal{D}$, and then use the scenario approach to derive formal probabilistic guarantees that $\mathcal{S}_\mathcal{D}$ is simulated by $\mathcal{S}_\mathcal{T}$. Algorithm 1 constructs the transition system $\mathcal{S}_\mathcal{T}$ by, in summary:

1) associating a discrete state of $\mathcal{S}_\mathcal{T}$ to each leaf of $\mathcal{T}$ (line 6);
2) assigning a transition $q_{j_1} \to q_{j_2}$ if the minimum distance between any two points belonging to the projection of the hyper-rectangle R_{j_1} associated to leaf ℓ_{j_1} (constructed on the basis of the affine predictive model $\alpha_{j_1}^*, \alpha_{j_1,0}^*$) and to the hyper-rectangle R_{j_2} (associated to leaf ℓ_{j_2}) is smaller than $h_{j_1}^*$ (line 7);
3) assigning as initial conditions of $\mathcal{S}_\mathcal{T}$ all those whose hyper-rectangular equivalence class intersects with the initial conditions of $\mathcal{D}$ (line 8). In Algorithm 1 we denote, for the j-th leaf, by $\{V_{j,\nu}\}_{\nu=1}^{2^m}$ the set of vertices of the hyper-rectangle R_j.

The following Proposition 1 derives probabilistic guarantees to the event that the abstraction $\mathcal{S}_\mathcal{T}$ is not a simulation

Algorithm 1 Compute finite transition system model $\mathcal{S}_\mathcal{T}$

1: **Input:** $\mathcal{T}, \{\alpha_j^*, \alpha_{j,0}^*, h_j^*\}_{j=1}^{|\mathcal{T}|}$
2: **Output:** $\mathcal{S}_\mathcal{T} = (Q_\mathcal{T}, \varnothing, \to_\mathcal{T}, Q_\mathcal{T}^0)$
3: **for all** $j_1, = 1, \dots, |\mathcal{T}|, j_2, = 1, \dots, |\mathcal{T}|$ **do**
4: Compute the distance $\Pi(j_1, j_2)$ by solving:

$$\Pi(j_1, j_2) = \min_{x_{j_1}, x_{j_2}} ||x_{j_1} - x_{j_2}||_2^2 \quad (11)$$

subject to:

$$x_{j_1} = \sum_{\nu=1}^{2^m} \gamma_{j_1,\nu} \left(\alpha_{j_1}^* V_{j_1,\nu} + \alpha_{j_1,0}^*\right) \quad (12)$$
$$\sum_{\nu=1}^{2^m} \gamma_{j_1,\nu} = 1, \quad 0 \le \gamma_{j_1,\nu} \le 1, \forall \nu = 1, \dots, 2^m \quad (13)$$
$$x_{j_2} = \sum_{\nu=1}^{2^m} \gamma_{j_2,\nu} V_{j_2,\nu} \quad (14)$$
$$\sum_{\nu=1}^{2^m} \gamma_{j_2,\nu} = 1, \quad 0 \le \gamma_{j_2,\nu} \le 1, \forall \nu = 1, \dots, 2^m \quad (15)$$

5: **end for**
6: Define $Q_\mathcal{T} = \{q_1, \dots, q_{|\mathcal{T}|}\}$
7: Define $q_{j_1} \to_\mathcal{T} q_{j_2} \Leftrightarrow \Pi(j_1, j_2) \le h_{j_1}^*$
8: Define $Q_\mathcal{T}^0 = \{q_j \in Q_\mathcal{T} : R_j \cap X^0 \neq \varnothing\}$
9: **Return** $\mathcal{S}_\mathcal{T} = (Q_\mathcal{T}, \varnothing, \to_\mathcal{T}, Q_\mathcal{T}^0)$

of the concrete system $\mathcal{D}$, i.e. that some trajectories of the concrete system that are not present in the training dataset invalidate the simulation relation between $\mathcal{S}_\mathcal{T}$ and $\mathcal{S}_\mathcal{D}$.

Proposition 1: Let Algorithms 0 and 1 be applied to a dataset $(\mathcal{X}, \mathcal{Y})$, then $\mathcal{P}^N\{\mathcal{P}\{\mathcal{S}_\mathcal{D} \not\preceq \mathcal{S}_\mathcal{T}\} > \epsilon\} \le \beta$ for any given $\beta \in (0, 1)$, with $\epsilon = \max_{j=1,\dots,|\mathcal{T}|} \min\{\epsilon_{j1}, \epsilon_{j2}\}$ and

$$\epsilon_{j1} = \begin{cases} 1 & k_j^* = |\ell_j|, \\ 1 - \left(\frac{\beta}{|\ell_j| \binom{|\ell_j|}{k_j^*}}\right)^{(|\ell_j| - k_j^*)^{-1}} & \text{otherwise.} \end{cases} \quad (16)$$

$$\beta = \sum_{i=0}^{n(m+1)} \binom{|\ell_j|}{i} (\epsilon_{j2})^i (1 - \epsilon_{j2})^{|\ell_j| - i}. \quad (17)$$

Proof: Consider the output of Algorithm 0: by [15, Proposition 1] the following holds $\forall x_j \in R_j, j \in \{1, \dots, |\mathcal{T}|\}$:

$$\mathcal{P}^N\{\mathcal{P}\{||f(x_j) - \mathcal{T}(x_j)||^2 \ge h_j^*\} > \epsilon\} \le \beta. \quad (18)$$

for the worst-case probability of violation among all leaves $\epsilon = \max_{j=1,\dots,|\mathcal{T}|} \epsilon_j$, with ϵ_j the least conservative bound derived via the nonconvex and convex cases conditions applied to each leaf ℓ_j, namely with ϵ_{j1} defined by (16) and ϵ_{j2} as in (17). Note that since the cardinality of the optimization variables of (9) is equal to $n(m+1)+1$, then differently from [15, Proposition 1] the dimension d (using the same notation as in (4)) to derive (17) is given by $d = n(m+1)+1$.

Consider now any $x_{j_1} \in R_{j_1}, j_1 \in \{1, \dots, |\mathcal{T}|\}$. First define the set $\mathcal{P}_\mathcal{T}(R_{j_1}) \doteq \{\alpha_{j_1}^* x + \alpha_{j_1,0}^*, x \in R_{j_1}\}$ the

projection of all states in R_{j_1} through the affine function $\alpha^*_{j_1}, \alpha^*_{j_1,0}$, which can be defined by the constraints (12), (13): it is well known (see e.g. [23]) that $\mathcal{P}_{\mathcal{T}}(R_{j_1})$ is a polytope, not necessarily a hyper-rectangle.

Then, define the set $\mathcal{P}_{\mathcal{D}}(R_{j_1}) \doteq \{f(x), x \in R_{j_1}\}$ the projection of all states in R_{j_1} through the dynamics $f(\cdot)$ of $\mathcal{D}$: clearly, $\mathcal{P}_{\mathcal{D}}(R_{j_1})$ is not necessarily a polytope.

Consider now any $j_2 \in \{1, \ldots, |\mathcal{T}|\}$ and note that constraints (14), (15) define the hyper-rectangle R_{j_2}. $\Pi(j_1, j_2)$, computed by solving the QP (11), is the smallest Euclidean distance between any pair (x_{j_1}, x_{j_2}), where x_{j_1} is any point of the polytope $\mathcal{P}_{\mathcal{T}}(R_{j_1})$ and x_{j_2} is any point of the hyper-rectangle R_{j_2}. By (18) and the definition of $\Pi(j_1, j_2)$ it follows that $\forall j_1, j_2 \in \{1, \ldots, |\mathcal{T}|\}$

$$\mathcal{P}^N\{\mathcal{P}\{(x \in R_{j_1}) \wedge (f(x) \in R_{j_2}) \wedge (\alpha^*_{j_1} x + \alpha^*_{j_1,0} \notin R_{j_2})\} > \epsilon\} \leq \beta. \tag{19}$$

Let us now define our candidate simulation relation $\Gamma \doteq \{(x, q_j) : x \in R_j, j \in \{1, \ldots, |\mathcal{T}|\}\} \subseteq \mathbb{R}^m \times Q_{\mathcal{T}}$. The definition of $\rightarrow_{\mathcal{T}}$ in line 7 of Algorithm 1, together with Equation (19), provide probabilistic bounds to the event that, for any pair $x \in R_{j_1}, x' \in R_{j_2}$ there exists $x \rightarrow_{\mathcal{D}} x'$ (i.e. $x' = f(x)$, but not $q_{j_1} \rightarrow q_{j_2}$: namely, Equation (19) provides probabilistic bounds to the event that Γ does not satisfy the conditions in Definition 2, even considering trajectories (scenarios) of the concrete system $\mathcal{D}$ that are not sampled in the training dataset.

The conditions in Definition 3 are always satisfied since, by line 8 of Algorithm 1, for any $x \in X^0$, $x \in R_j$, there exists $q_j \in Q^0_{\mathcal{T}}$ with $(x, q_j) \in \Gamma$. This concludes the proof. ■

In the following section we will validate our algorithm and probabilistic bounds over a benchmark.

IV. Validation via benchmark

To validate our methodology, we chose the navigation benchmark proposed in [17] and utilized e.g. in [24], [25], [26], [27] for abstraction or verification of a hybrid system.

A. Navigation Benchmark model

This benchmark models the movement of an object such as car or robot on a 2D grid environment. In $\mathbb{R}^2$, the object can move through the 3×3 grid shown in Figure 1, where the size of each grid is 1 in width and height. The target of the object is to reach a specific grid A while avoiding grid B or exiting the grid. Let $\mathbf{s} \in \mathbb{R}^2$ be the position and $\mathbf{v} \in \mathbb{R}^2$ be the velocity of the object: the dynamics are defined by an ordinary differential equation $\dot{\mathbf{v}} = \mathbf{A}(\mathbf{v} - \mathbf{v}_d)$, where $\mathbf{A} \in \mathbb{R}^{2\times 2}$ (20) has eigenvalues with strictly negative real part to assure that the velocity will converge to the desired one

$$A = \begin{pmatrix} -1.2 & 0.1 \\ 0.1 & -1.2 \end{pmatrix}. \tag{20}$$

The desired velocity $\mathbf{v}_d$ depends on the cell according to the position of the object in the 3×3 grid and can be calculated by $\mathbf{v}_d = [\sin\left(i \cdot \frac{\pi}{4}\right), \cos\left(i \cdot \frac{\pi}{4}\right)]', i \in \{2, 3, 4\}$.

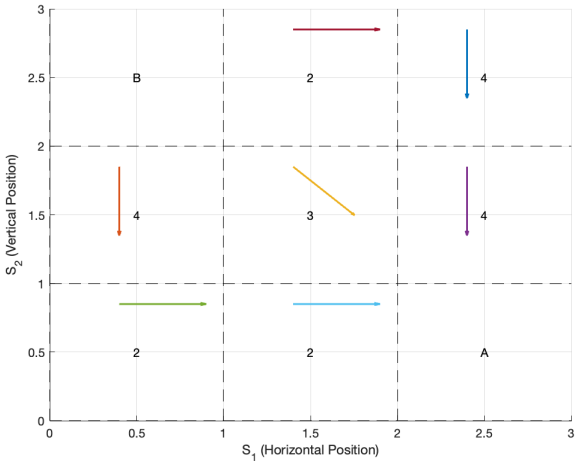

Fig. 1: Navigation benchmark [17] 3×3 grid environment.

B. Dataset Generation

To generate the data we simulated the benchmark system in Matlab while using the following configurations. Our state $x = [\mathbf{s}', \mathbf{v}']' \in \mathbb{R}^4$, consisting of position $\mathbf{s} \in \mathbb{R}^2$ and velocity $\mathbf{v} \in \mathbb{R}^2$. The initial position of the object was randomly sampled via uniform distribution within the grid excluding the target cell A and the forbidden cell B, and the initial velocity was randomly sampled via uniform distribution from the interval $[-\bar{\mathbf{v}}, \bar{\mathbf{v}}] \times [-\bar{\mathbf{v}}, \bar{\mathbf{v}}]$. We simulated system (20) discretised with sampling time $T = 0.01s$ and generated N one-step trajectories $\{x_i(0), x_i(1)\}_{i=1}^N$, which we split into 80% for training and 20% for testing.

C. Transition system construction and discussion

For our simulations we have used a virtual machine equipped with 16 CPUs and 64GB of RAM, and exploited Matlab, Python, Parallel Computing Toolbox, Pandas Scikit-Learn, Numpy and CVXPY. We applied Algorithms 0 and 1 to our dataset of trajectories configuring the stopping time of the CART algorithm via the maximum number of leaves, which has been tuned considering the tradeoff between accurate predictive model and reasonable probabilistic guarantees. We used NRMSE% as performance metric for the predictive accuracy.

1) Simulation 1: In the first test of our validation we considered a smaller set of initial conditions, with $\bar{\mathbf{v}} = 0.2$, and tuned the parameter *MaxLeafNodes*$= 150$. The execution time for both Algorithm 0 and Algorithm 1 was $\sim$ 14 minutes. We can see from the Table I that the CART algorithm and the predictive model performed very accurately, achieving 0.0240% NRMSE% on the test data. The algorithm generated a state space partition, whose projection on the 3D space $(\mathbf{s}_1, \mathbf{v}_1, \mathbf{v}_2)$ is illustrated in Figure 2. A transition system with 150 states and 3484 transitions is illustrated in Figure 3, which is calculated using Algorithm 1. We remark that the partition is strongly non-uniform, i.e. using the smallest sizes of each hyper-rectangle we would need 2304 equivalence classes to cover the investigated state-space.

Figure 3 provides a screenshot of the abstraction transition system, where the unique red state (bottom-left of the figure) represents a sink state corresponding to a trajectory that came out of the grid, and the 18 yellow states represent equivalence classes of states that can possibly exit the grid.

Size of Dataset	$N = 10^7$
Training / Testing Dataset split	80% / 20%
MaxLeafNodes	150
NRMSE% testing	0.0240%
Number of states via Algorithm 1	150
Number of states via uniform partitioning	2304
Number of Transitions via Algorithm 1	3484
Probabilistic bounds via Proposition 1	$\epsilon = 1,01 \cdot 10^{-3}, \beta = 10^{-6}$

TABLE I: Simulation 1, table of parameters and results

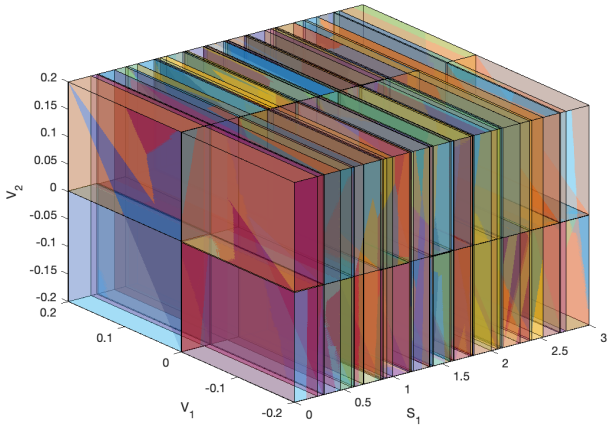

Fig. 2: Simulation 1, 3D projection of state space partition

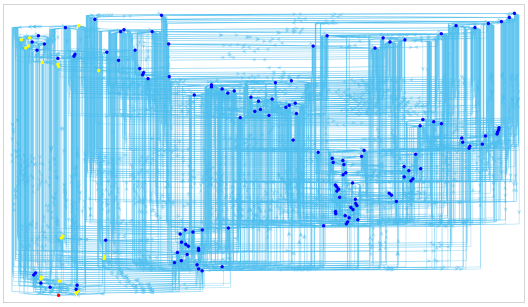

Fig. 3: Simulation 1, Transition system finite abstraction

Applying our probabilistic guarantees with $\beta = 10^{-6}$ we found out that $\epsilon = 1,01 \cdot 10^{-3}$, characterizing quantitatively the reliability of our abstraction by Proposition 1: more precisely, these values bound on the probability that the abstraction is a simulation of the original system also for trajectories that are not present in the training dataset: in our simulation it has been verified that the trajectories of the testing dataset never violate the simulation relation between the original system and the calculated abstraction.

2) Simulation 2: In the second test of our validation we considered an intermediate set of initial conditions, with $\bar{\mathbf{v}} = 0.5$, and tuned the parameter *MaxLeafNodes*$= 100$. The execution time for both Algorithm 0 and Algorithm 1 was ~ 14 minutes. We can see from the Table II that the CART algorithm and the predictive model performed very accurately, achieving 0.0293% NRMSE% on the test data. The algorithm generated a state space partition, whose projection on the 3D space $(\mathbf{s}_1, \mathbf{v}_1, \mathbf{v}_2)$ is illustrated in Figure 4. A transition system with 100 states and 3927 transitions is illustrated in Figure 5, which is calculated using Algorithm 1.

Size of Dataset	$N = 10^7$
Training / Testing Dataset split	80% / 20%
MaxLeafNodes	100
NRMSE% testing	0.0293%
Number of states via Algorithm 1	100
Number of states via uniform partitioning	441
Number of Transitions via Algorithm 1	3927
Probabilistic bounds via Proposition 1	$\epsilon = 1,01 \cdot 10^{-3}, \beta = 10^{-6}$

TABLE II: Simulation 2, table of parameters and results

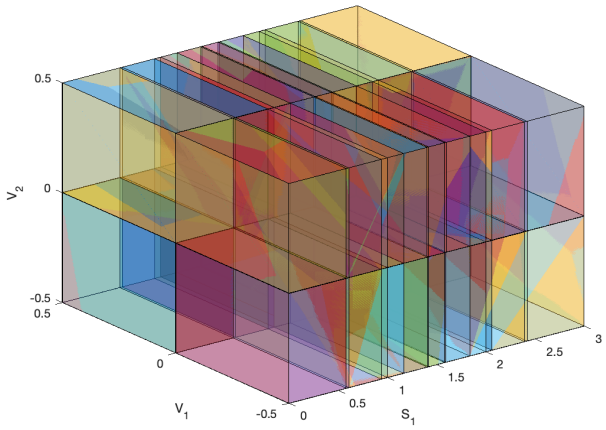

Fig. 4: Simulation 2, 3D projection of state space partition

We remark that the partition is strongly non-uniform, i.e. using the smallest sizes of each hyper-rectangle we would need 441 equivalence classes to cover the investigated state-space. Figure 5 provides a screenshot of the abstraction transition system, where the unique red state (bottom-left of the figure) represents a sink state corresponding to a trajectory that came out of the grid, and the 14 yellow states represent equivalence classes of states that can possibly exit the grid. Applying our probabilistic guarantees with $\beta = 10^{-6}$ we

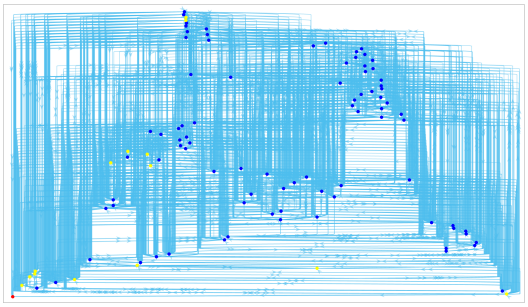

Fig. 5: Simulation 2, Transition system finite abstraction

found out that $\epsilon = 1,01 \cdot 10^{-3}$, characterizing quantitatively the reliability of our abstraction by Proposition 1. Also in Simulation 2 it has been verified that the trajectories of the testing dataset never violate the simulation relation between the original system and the calculated abstraction.

3) Simulation 3: In the third test of our validation we considered a larger set of initial conditions, with $\bar{\mathbf{v}} = 1$, and tuned the parameter *MaxLeafNodes*$= 350$. The execution time for both Algorithm 0 and Algorithm 1 was ~ 14 minutes. We can see from the Table III that, also in this case, the CART algorithm and the predictive model performed very accurately, achieving 0.02610% NRMSE% on the test data. The algorithm generated a state space partition, whose projection on the 3D space $(\mathbf{s}_1, \mathbf{v}_1, \mathbf{v}_2)$ is illustrated in Figure 6. A transition system with 350 states and 16858 transitions is illustrated in Figure 7, which is calculated using Algorithm 1. We remark that also in Simulation 3 the partition is strongly non-uniform, i.e. using the smallest sizes of each hyper-rectangle we would need 1225 equivalence classes to cover the investigated state-space. Figure 7 provides a screenshot of the abstraction transition system, where the unique red state (bottom-center of the figure) represents a sink state corresponding to a trajectory that came out of the grid, and the 52 yellow states represent equivalence classes of states that can possibly exit the grid. Applying our probabilistic guarantees with $\beta = 10^{-6}$ we found out that $\epsilon = 1,5 \cdot 10^{-3}$, characterizing quantitatively the reliability of our abstraction by Proposition 1. Also in Simulation 3 it has been verified that the trajectories of the testing dataset never violate the simulation relation between the original system and the calculated abstraction.

As a final remark, we recall that the scenario approach requires only the assumption that the samples (i.e. the scenarios) are drawn independently: as well known from the scientific literature, such generality is paid by conservatism of the probabilistic bounds. The statistical bounds validated in our simulations are thus coherent with this remark. Also, increasing the volume of the set of initial conditions the number of discrete states of our abstraction scales well, and remains much smaller than the cardinality of a uniform partition. We expect that our algorithm also scales quite well increasing the dimension m of the state space $\mathbb{R}^m$: this is venue for future research.

Size of Dataset	$N = 10^7$
Training / Testing Dataset split	80% / 20%
MaxLeafNodes	350
NRMSE% testing	0.02610%
Number of states via Algorithm 1	350
Number of states via uniform partitioning	1225
Number of Transitions via Algorithm 1	16858
Probabilistic bounds via Proposition 1	$\epsilon = 1,5 \cdot 10^{-3}, \beta = 10^{-6}$

TABLE III: Simulation 3, table of parameters and results

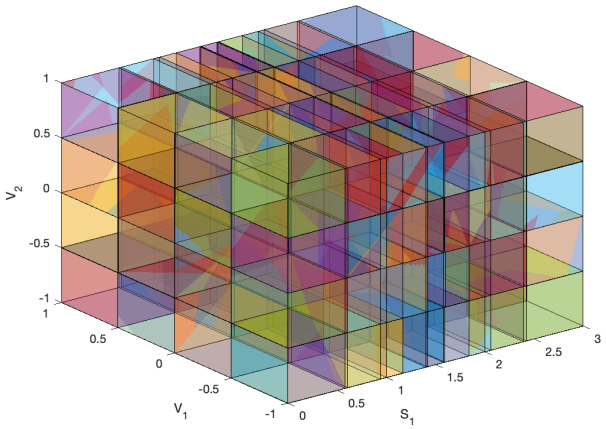

Fig. 6: Simulation 3, 3D projection of state space partition

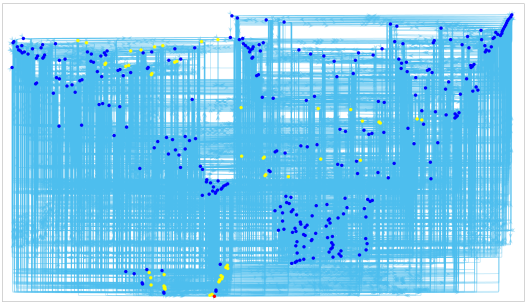

Fig. 7: Simulation 3, Transition system finite abstraction

V. Conclusions

In this paper we proposed a novel data driven methodology to construct a finite abstraction of a dynamical system by means of a transition system, proved that our abstraction is a *simulation* of the original dynamical system with quantified probabilistic guarantees, and validated on a benchmark on hybrid systems that it is very promising for handling abstractions of large dimensional systems. In future work we plan to

extend our theoretical results to the notions of approximate and alternating (bi)simulation, and stress test our algorithm on benchmarks with very large state space dimension and on real datasets, e.g. as in [28], [29].

REFERENCES

[1] Y. Zacchia Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *J. Syst. Softw.*, vol. 149, pp. 174–216, 2019.

[2] D. Ding, Q.-L. Han, X. Ge, and J. Wang, "Secure state estimation and control of cyber-physical systems: A survey," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 1, pp. 176–190, 2020.

[3] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, no. 2, pp. 971–984, Jul. 2000.

[4] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. The MIT Press, Cambridge, Massachusetts, 2002.

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

[6] J. Calbert, A. Girard, and R. M. Jungers, "Classification of simulation relations for symbolic control," 2024. [Online]. Available: https://arxiv.org/abs/2410.06083

[7] R. Coppola, A. Peruffo, and M. Mazo Jr., "Data-driven abstractions for verification of deterministic systems," *IEEE Control Syst. Lett.*, vol. 7, pp. 115–120, 2023.

[8] A. Lavaei and E. Frazzoli, "Data-driven synthesis of symbolic abstractions with guaranteed confidence," *IEEE Control Syst. Lett.*, vol. 7, pp. 253–258, 2022.

[9] A. Banse, L. Romao, A. Abate, and R. M. Jungers, "Data-driven memory-dependent abstractions of dynamical systems," in *Proc. Mach. Learn. Res.*, vol. 211, 2023, pp. 891–902.

[10] A. Makdesi, A. Girard, and L. Fribourg, "Data-driven abstraction of monotone systems," in *Proc. Mach. Learn. Res.*, vol. 144, 2021, pp. 803–814.

[11] A. Abate, M. Giacobbe, and Y. Schnitzer, "Bisimulation learning," in *Proc. Int. Conf. Comput. Aided Verification (CAV)*, ser. Lecture Notes Comput. Sci., vol. 14683. Springer, 2024, pp. 161–183.

[12] D. Ajeleye, A. Lavaei, and M. Zamani, "Data-driven controller synthesis via finite abstractions with formal guarantees," *IEEE Control Syst. Lett.*, vol. 7, pp. 3453–3458, 2023.

[13] S. Sadraddini and C. Belta, "Formal guarantees in data-driven model identification and control synthesis," in *Proc. Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, 2018, pp. 147–156.

[14] F. Smarra, G. D. Di Girolamo, V. De Iuliis, A. Jain, R. Mangharam, and A. D'Innocenzo, "Data-driven switching modeling for mpc using regression trees and random forests," *Nonlinear Anal.: Hybrid Syst.*, vol. 36, 2020, Art. no. 100882.

[15] A. D'Innocenzo and F. Smarra, "Learning piecewise ARX models via regression trees with probabilistic guarantees," in *Proc. Eur. Control Conf. (ECC)*, 2024, pp. 2083–2088.

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

[17] A. Fehnker and F. Ivančić, "Benchmarks for hybrid systems verification," in *Proc. Int. Workshop Hybrid Syst.: Comput. Control*. Springer, 2004, pp. 326–341.

[18] G. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, Dec. 2003.

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

[20] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[21] S. K. Murthy, S. Kasif, and S. Salzberg, "Multivariate decision trees," *Mach. Learn.*, vol. 18, no. 1, pp. 63–87, 1994.

[22] P. Ciampi, A. Giordani, and L. Politi, "Multivariate data analysis through classification and regression trees," *Statistical Methods Appl.*, vol. 17, no. 3, pp. 225–250, 2008.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004. [Online]. Available: https://stanford.edu/ boyd/cvxbook/

[24] R. Alur, T. Dang, and F. Ivančić, "Predicate abstraction for reachability analysis of hybrid systems," *ACM Trans. Embedded Comput. Syst.*, vol. 5, no. 1, pp. 152–199, 2006.

[25] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle, "Abstraction-based guided search for hybrid systems," in *Proc. Int. Symp. Model Checking Softw. (SPIN)*, ser. Lecture Notes Comput. Sci., vol. 7976. Springer, 2013, pp. 117–134.

[26] S. Sankaranarayanan and A. Tiwari, "Relational abstractions for continuous and hybrid systems," in *Proc. Int. Conf. Comput. Aided Verification (CAV)*, ser. Lecture Notes Comput. Sci., vol. 6806. Springer, 2011, pp. 686–702.

[27] G. Frehse, "Phaver: algorithmic verification of hybrid systems past hytech," *Int. J. Softw. Tools Technol. Transfer*, vol. 10, pp. 263–279, 2008.

[28] F. Smarra, A. Jain, T. De Rubeis, D. Ambrosini, A. D'Innocenzo, and R. Mangharam, "Data-driven model predictive control using random forests for building energy optimization and climate control," *Appl. Energy*, vol. 226, pp. 1252–1272, 2018.

[29] F. Smarra, J. Tjen, and A. D'Innocenzo, "Learning methods for structural damage detection via entropy-based sensors selection," *Int. J. Robust Nonlinear Control*, vol. 32, no. 10, pp. 6035–6067, 2022.