Learning-based Symbolic Abstractions for Nonlinear Control Systems ⋆

Kazumune Hashimoto [a], Adnane Saoud [b], Masako Kishida [c], Toshimitsu Ushio [d], Dimos V. Dimarogonas [e]

[a] *Graduate School of Engineering, Osaka University, Suita, Japan*

[b] *Laboratoire des Signaux et Systèmes, Université Paris-Saclay, CNRS, CentraleSupélec*

[c] *National Institute of Informatics (NII), Tokyo, Japan.*

[d] *Graduate School of Engineering and Science, Osaka University, Toyonaka, Japan*

[e] *School of Electrical Engineering, KTH Royal Institute of Technology, 10044 Stockholm, Sweden.*

Abstract

Symbolic models or abstractions are known to be powerful tools for the control design of cyber-physical systems (CPSs) with logic specifications. In this paper, we investigate a novel *learning-based* approach to the construction of symbolic models for nonlinear control systems. In particular, the symbolic model is constructed based on *learning* the un-modeled part of the dynamics from training data based on state-space exploration, and the concept of an alternating simulation relation that represents behavioral relationships with respect to the original control system. Moreover, we aim at achieving *safe exploration*, meaning that the trajectory of the system is guaranteed to be in a safe region for all times while collecting the training data. In addition, we provide some techniques to reduce the computational load, in terms of memory and computation time, of constructing the symbolic models and the safety controller synthesis, so as to make our approach practical. Finally, a numerical simulation illustrates the effectiveness of the proposed approach.

Key words: Symbolic models, uncertain systems, safety controller synthesis, Gaussian Processes

1 Introduction

In cyber-physical systems (CPS), computational devices are tightly integrated with physical processes. Embedded computers monitor the behavior of the physical processes through sensors, and usually control them through actuators using feedback loops. Nowadays, CPSs are ubiquitous in modern control engineering, including automobiles, aircraft, building control systems, chemical plants, transportation systems, and so on. Many CPSs are safety critical or mission critical: it must ensure that the system operates correctly meeting the satisfaction of safety or some desired specifications. Formal methods are known to provide essential tools for the design of CPSs, as they give theoretical or rigorous mathematical *proofs* that the system works correctly meeting the desired specification [1]. While the formal methods have been originally developed in software engineering that aims at finding bugs or security vulnerabilities in the software, the methodologies have been recently recognized to be useful in other applications, including the control design of CPSs. In particular, one of the most successful methods that interface the formal methods and the control design of CPSs is the so-called *symbolic control*, see, e.g., [2]. The main objective of the symbolic control is to design controllers for CPSs with logic specifications (as detailed below). In such approaches, *symbolic models* or *abstractions* are constructed based on the original control systems. Roughly speaking, while the original control system is represented in a *continu-*

⋆ This work is supported by JST ERATO Grant Number JPMJER1603, and by JST CREST Grant Number JPMJCR2012, Japan.

Email addresses: hashimoto@eei.eng.osaka-u.ac.jp (Kazumune Hashimoto), Adnane.Saoud@l2s.centralesupelec.fr (Adnane Saoud), kishida@nii.ac.jp (Masako Kishida), ushio@sys.es.osaka-u.ac.jp (Toshimitsu Ushio), dimos@kth.se (Dimos V. Dimarogonas).

ous state (and input) space, the symbolic model is represented in a *discrete* state (and input) space, while preserving the behavior of the original control system. As such, controllers can be designed based on several algorithmic techniques from supervisory control of discrete event systems, such as a safety/reachability game [3].

The symbolic approach is known to be a powerful tool for the control design of CPSs in the following three ways. First, it allows us to synthesize controllers for general nonlinear dynamical systems with state and input constraints. Second, by constructing the symbolic model, we can take into account the constraints that are imposed on the cyber part with regard to the digital platform, such as a quantization effect. Third, it allows us to synthesize controllers under various control specifications, including safety, reachability, or more complex ones such as those expressed by linear temporal logic (LTL) formulas or automata on infinite strings. As previously mentioned, symbolic models or abstractions are constructed such that they are represented in a discrete state space while preserving the behavior of the original control system. More formally, behavioural relationships such as the concept of approximate (bi-)simulation relation, see, e.g., [3,4,5,6,7,8,9,10,11,12,13], are used to relate the behaviours of the original control system and its symbolic model. For example, [5] employs an approximate bisimulation relation to construct the symbolic model for nonlinear, incrementally asymptotically stable systems. [8] employs an approximate alternating simulation relation, so that the symbolic models can be constructed for general (incrementally forward complete) nonlinear systems without any assumption on stability. Moreover, [11,12] characterize the notion of robustness for input-output dynamically stable systems based on the concept of a contractive approximate simulation relation.

In this paper, we focus on investigating the construction of symbolic models for nonlinear control systems. In particular, we consider the case where the dynamics of the plant includes state-dependent, *un-modeled* dynamics. In contrast to the aforecited abstraction schemes, we propose a *learning-based* solution to this problem, in which the symbolic model is constructed based on learning the un-modeled dynamics from training data. Moreover, we aim at achieving *safe-exploration*, meaning that the trajectory of the system stays inside a safe set for all times while collecting the training data. Achieving safe exploration is particularly useful for safety critical CPSs, see, e.g., [14]. More technically, as a starting point of our approach, we employ the Gaussian process (GP) regression [15] in order to estimate the un-modeled dynamics from training data. As we will see later, it is shown that, under some smoothness assumption on the un-modeled dynamics, an error bound on the un-modeled dynamics can be derived based on the result from [16]. Note that, in contrast to previous approaches of learning-based controller synthesis with the GP regression (e.g., [17,18]) that make use of an error (or regret) bound that involves an information gain, here we will make use of a deterministic error bound that does not involve the information gain, which has been also derived in [16] (for details, see Lemma 2 and Remark 2 in this paper). Based on this error bound and the concept of an approximate alternating simulation relation [8], we then provide an approach to construct the symbolic model. To achieve the safe exploration, we also provide a safety controller synthesis via a safety game [3]. Finally, we provide an overall algorithm that collects the training data from scratch and constructs the symbolic model. Along with this algorithm, we provide several techniques to reduce the computational load of constructing the symbolic model and the safety controller synthesis. In particular, we provide a lazy abstraction scheme, in which the transitions of the symbolic model are updated only around the region where the training data is collected.

(Related works): The approach presented in this paper is related to previous literature in terms of symbolic control (or temporal logics) and controller synthesis for dynamical systems learned by training data. In what follows, we discuss how our approach differs from previous works and highlight our main contributions.

As previously mentioned, there have been a wide variety of symbolic control techniques for dynamical systems, e.g., [4,5,6,7,8,19,20,13]; however, most of the previous approaches typically assume that the dynamics of the plant is completely *known* or they consider uniform disturbance that is not learned from training data. The learning-based approach is advantageous over the uniform disturbance-based approach in the following sense. In the uniform disturbance-based approach, every transition of the symbolic model is defined by taking the *worst case effect* of the un-modeled function, since the un-modeled function will not be learned from data. On the other hand, in the learning-based approach, the unmodeled function will be learned and thus its uncertainty will decrease as the state exploration progresses. Hence, the symbolic model will have fewer redundant transitions than the uniform disturbance-based approach, and this leads to obtaining a larger region that guarantees safety (i.e., controlled invariant set). To the best of our knowledge, there are only few works of symbolic or temporal logic control for a dynamical system that is partially unknown and is learned by training data (e.g., by the GP regression) [21,22]. In [21], the authors provided a way to obtain a finite abstraction using interval Markov decision processes (IMDPs) with the unknown dynamics learned by the GP regression. The abstraction has been then utilized for safety verification. Our approach is different from this previous work in the following sense: first, while the proposed approach in [21] makes it possible to provide probabilistic guarantees, in this paper we are able to provide deterministic guarantees. Second, while in [21] the symbolic model is constructed to deal with only safety specifications, in this paper, we are constructing the symbolic abstraction in the more general

sense of alternating simulation relations, i.e., we can refine a controller for the symbolic model into a controller for the original system for any specification, and not just safety. Finally, for the particular class of safety specifications, the authors in [21] provided a way of constructing a symbolic model with *given initial* training data, and they did not provide an approach to update the symbolic model when new training data are collected online, and which is the main issue in learning-based control, since the objective is to exploit the new training data collected online. In contrast, our approach provides a new computationally efficient approach to collect new training data while reducing the computation load to update the symbolic model and to synthesize safety controllers. In [22], the authors provided a way to detect faults using signal temporal logic (STL) for partially unknown dynamical systems and these are learned by the GP regression. However, the problem setup considered in [22] is different from the one considered in this paper. Specifically, while [22] considered a *monitoring* scheme in which they monitor behaviors of the system without control inputs and check if a given STL formula is satisfied, this paper considers a *synthesis* scheme in which we find a controller to satisfy certain specifications (expressed by, e.g., temporal logic formulas). The proposed approach is also different; while our approach aims at constructing symbolic models from training data while guaranteeing a given specification, the approach in [22] provided a monitoring scheme by employing a robustness degree of STL formulas.

Apart from the use of symbolic control, various learning-based controller synthesis techniques with the GP regression have been proposed. Most of the previous works aim at synthesizing controllers to achieve stability/tracking [23,24,25,26], or to guarantee safety [17,18,27,28,29,30,31,32,33]. Since we here construct a safety controller to achieve a safe exploration, our approach is particularly related to the second category, i.e., [17,18,27,28,29,30,31,32,33]. For example, the authors in [17] (resp. [18]) proposed an approach to learn a region of attraction (ROA) using safety controllers for continuous-time systems: $\dot{x}(t) = f(x(t), u(t)) + g(x(t), u(t))$ (resp. discrete-time systems: $x(k+1) = h(x(k), u(k)) + g(x(k), u(k))$), where the function $g(\cdot)$ is unknown and it is learned by the GP regression. The assumptions on the unknown function $g(\cdot)$ that are made in [17] and [18] are the same as the ones we are using in this paper, namely the fact that the unknown function lies in the reproducing kernel Hilbert space (RKHS). However, the authors in [17,18] assumed the existence of a *known* Lyapunov function for the nominal system $\dot{x}(t) = f(x(t), u(t))$ ($x(k+1) = h(x(k), u(k))$), for which the computation may be difficult for general nonlinear systems. The approach presented in this paper allows us to deal with more general complex specifications (including safety) without requiring the existence of a Lyapunov function. The approaches presented in [30,31,32] used a control barrier function and [29] used a Hamilton-Jacobi-Issac (HJI) equation to synthesize safety controllers with the GP regression. The proposed approach presented in this paper is significantly different from [30,31,32,29] in the following sense. First, note that while the goal of the previous work is to derive a safety controller, our *main* goal is to construct a symbolic model. Constructing the symbolic model is beneficial since it allows not only to compute a safety controller, but also controllers from more general complex specifications, such as those expressed by temporal logic specifications and automata on infinite strings. Moreover, while in [30,31] (resp. [32]), the use of barrier functions makes it only possible to deal with the class of polynomial dynamical systems (resp. input affine systems), the proposed approach in this paper makes it possible to deal with general nonlinear systems, and this is achieved by employing the symbolic models. Besides, while solving the HJI equation generally requires a heavy computational load, [29] did not provide a way of speeding up the computation of solving the HJI equation when a new set of training data is obtained. On the other hand, we here propose a way of reducing the computation load to update the symbolic model as well as synthesize safety controllers even if a new set of training data is obtained online.

Due to page limitation, some detailed proofs, remarks, and simulation results are given in the extended version of this paper [34].

Notation. Let $\mathbb{N}$, $\mathbb{N}_{\geq a}$, $\mathbb{N}_{>a}$, $\mathbb{N}_{a:b}$ be the sets of integers, integers larger than or equal to a, integers larger than a, and integers from a to b respectively. Let $\mathbb{R}$, $\mathbb{R}_{\geq a}$, $\mathbb{R}_{>a}$ be the sets of reals, reals larger than or equal to a and reals larger than a, respectively. Given $a, b \in \mathbb{R}$ with $a \leq b$, let $[a, b]$ be the interval set from a to b. Given $a, b \in \mathbb{R}_{\geq 0}$, we let $[a \pm b] = [a - b, a + b]$. Denote by $\|x\|_\infty$ the infinity norm of a vector x. Given $x \in \mathbb{R}^n, \varepsilon \in \mathbb{R}_{\geq 0}$, let $\mathcal{B}_\varepsilon(x) \subset \mathbb{R}^n$ be the ball set given by $\mathcal{B}_\varepsilon(x) = \{x \in \mathbb{R}^n \mid \|x\|_\infty \leq \varepsilon\}$. Given $\mathcal{X} \subseteq \mathbb{R}^n$ and $\eta > 0$, denote by $[\mathcal{X}]_\eta \subset \mathbb{R}^n$ the lattice in $\mathcal{X}$ with the quantization parameter η, i.e., $[\mathcal{X}]_\eta = \{x \in \mathcal{X} \mid x_i = a_i \eta,\ a_i \in \mathbb{N},\ i = 1, 2, \ldots, n\}$, where $x_i \in \mathbb{R}$ is the i-th element of x. Given $x \in \mathbb{R}^n$, $\mathcal{X} \subseteq \mathbb{R}^n$, denote by $\mathsf{Nearest}_{\mathcal{X}}(x)$ the closest points in $\mathcal{X}$ to x, i.e., $\mathsf{Nearest}_{\mathcal{X}}(x) = \arg\min_{x' \in \mathcal{X}} \|x - x'\|_\infty$. Given $\mathcal{X} \subset \mathbb{R}^n$, we let $\mathsf{Interior}_\varepsilon(\mathcal{X}) = \{x \in \mathcal{X} \mid \mathcal{B}_\varepsilon(x) \subseteq \mathcal{X}\}$.

2 Preliminaries

In this section we recall some basic concepts of the Gaussian Process (GP) regression [15], transition systems and approximate alternating simulation relations [8].

2.1 Gaussian process regression

Consider a nonlinear function $h : \mathbb{R}^{n_x} \to \mathbb{R}$ perturbed by additive noise as $y = h(x) + v$, where

$x \in \mathbb{R}^{n_x}$ is the input, $y \in \mathbb{R}$ is the output, and $v \sim \mathcal{N}(0, \sigma^2)$ is the Gaussian distributed white noise. Given $m : \mathbb{R}^{n_x} \to \mathbb{R}$ and some kernel function $\mathsf{k} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$, suppose that, for any finite number of inputs $X = [x_1, \ldots, x_T]$ ($x_t \in \mathbb{R}^{n_x}$, $t \in \{1, \ldots, T\}$), the joint probability distribution of the corresponding outputs $y = [y_1, y_2, \ldots, y_T]^\mathsf{T}$ follows the multivariate Gaussian distribution: $y \sim \mathcal{N}(M, K)$, where $M = [m(x_1), \ldots, m(x_T)]$ and $K_{tt'} = \mathsf{k}(x_t, x_{t'})$, $t, t' \in \{1, \ldots, T\}$ ($K_{tt'}$ denotes the (t, t')-element of K). Then, we say that the function h follows a *Gaussian process* (GP) [15], and it is denoted by $h(x) \sim \mathcal{GP}(m(x), \mathsf{k}(x, x'))$.

In the GP *regression* problem, we start by assuming a GP prior: $h(x) \sim \mathcal{GP}(m(x), \mathsf{k}(x, x'))$. Let $\mathcal{D} = \{x_t, y_t\}_{t=1}^T$ denote a training data set. Then, using Bayes rule, the *posterior* distribution of the output for an arbitrary input $x \in \mathbb{R}^{n_x}$ follows the Gaussian distribution, i.e., $\Pr(y|x, \mathcal{D}) = \mathcal{N}(\mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D}))$. Here, the mean $\mu(x; \mathcal{D})$ and the variance $\sigma^2(x; \mathcal{D})$) are given by

$$\mu(x; \mathcal{D}) = m(x) + \mathsf{k}_T^{*\mathsf{T}}(x)(K + \sigma^2 I)^{-1}(Y - M), \quad (1)$$

$$\sigma^2(x; \mathcal{D}) = \mathsf{k}(x, x) - \mathsf{k}_T^{*\mathsf{T}}(x)(K + \sigma^2 I)^{-1}\mathsf{k}_T^*(x), \quad (2)$$

where I is the identity matrix of appropriate dimension, and $\mathsf{k}_T^*(x) = [\mathsf{k}(x, x_1), \ldots, \mathsf{k}(x, x_T)]^\mathsf{T}$.

2.2 Transition system, alternating simulation relation

We provide the notion of a transition system, which will be useful to describe a control system formalized later in this paper.

Definition 1 A *transition system* is a quadruple $S = (\mathcal{X}, x_0, \mathcal{U}, G)$, where:

- $\mathcal{X}$ is a set of states;
- $x_0 \in \mathcal{X}$ is an initial state;
- $\mathcal{U}$ is a set of inputs;
- $G : \mathcal{X} \times \mathcal{U} \to 2^{\mathcal{X}}$ is a transition map. □

Roughly speaking, we denote by $x' \in G(x, u)$ if and only if the system evolves from x to x' by applying the control input u. The state x' is called a *u-successor* of x. Moreover, we denote by $\mathcal{U}(x)$ the set of all inputs $u \in \mathcal{U}$, for which $G(x, u) \neq \varnothing$.

Next, we shall recall the notion of an *approximate alternating simulation relation*[7,8], which is a well-known concept to represent behavioral relationships on the similarity between two transition systems.

Definition 2 (ε-ASR) Let $S_a = (\mathcal{X}_a, x_{a0}, \mathcal{U}_a, G_a)$ and $S_b = (\mathcal{X}_b, x_{b0}, \mathcal{U}_b, G_b)$ be two transition systems. Given $\varepsilon \in \mathbb{R}_{\geq 0}$, a relation $R(\varepsilon) \subseteq \mathcal{X}_a \times \mathcal{X}_b$ is called an *ε-approximate Alternating Simulation Relation* (or ε-ASR for short) from S_a to S_b, if the following conditions are satisfied:

(C.1) $(x_{a0}, x_{b0}) \in R(\varepsilon)$;
(C.2) For every $(x_a, x_b) \in R(\varepsilon)$, we have $\|x_a - x_b\|_\infty \leq \varepsilon$;
(C.3) For every $(x_a, x_b) \in R(\varepsilon)$ and for every $u_a \in \mathcal{U}_a(x_a)$, there exist $u_b \in \mathcal{U}_b(x_b)$, such that the following holds: for every $x_b' \in G_b(x_b, u_b)$, there exists $x_a' \in G_a(x_a, u_a)$, such that $(x_a', x_b') \in R(\varepsilon)$. □

The transition system S_a serves as the *abstract* expression of S_b, in the sense that every transition of S_b can be *approximately simulated* by those of S_a according to (C.1)–(C.3) in Definition 2. The concept of an ε-ASR is particularly useful to synthesize a controller for the transition system S_b, based on the controller for S_a. That is, once we obtain S_a that guarantees the existence of an ε-ASR from S_a to S_b, we can synthesize a controller for S_b by *refining* a controller for S_a that can be synthesized by algorithmic techniques from discrete event systems, see, e.g., [3].

3 Problem formulation

In this section, we describe a control system that we seek to consider, provide the notion of a controlled invariant set, and describe the goal of this paper.

3.1 System description

Let us consider the following nonlinear systems:

$$x_{t+1} = f(x_t, u_t) + d(x_t) + v_t, \quad (3)$$

$$x_0 = \bar{x},\ u_t \in \mathcal{U},\ v_t \in \mathcal{V}, \quad (4)$$

for all $t \in \mathbb{N}_{\geq 0}$, where $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathbb{R}^{n_u}$ is the control input, $v_t \in \mathbb{R}^{n_x}$ is the additive noise, and $\bar{x} \in \mathbb{R}^{n_x}$ is the initial state. Moreover, $\mathcal{U} \subset \mathbb{R}^{n_u}$ and $\mathcal{V} \subset \mathbb{R}^{n_x}$ are the set of control inputs and the additive noise, respectively. It is assumed that $\mathcal{U}$ is compact and $\mathcal{V}$ is given by $\mathcal{V} = \{v \in \mathbb{R}^{n_x} \mid \|v\|_\infty \leq \sigma_v\}$ for a given $\sigma_v > 0$. Moreover, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the *known* function that captures the modeled (or nominal) dynamics, and $d : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is the state-dependent, *unknown* deterministic function that captures the un-modeled dynamics. Regarding the function f, we assume the following Lipschitz continuity:

Assumption 1 The function f is Lipschitz continuous in $x \in \mathbb{R}^{n_x}$, i.e., given $L_f \in \mathbb{R}_{\geq 0}$, $\|f(x_1, u) - f(x_2, u)\|_\infty \leq L_f \|x_1 - x_2\|_\infty$, $\forall x_1, x_2 \in \mathbb{R}^{n_x}, \forall u \in \mathcal{U}$. □

Regarding the unknown function d_i, $i \in \mathbb{N}_{1:n_x}$, in this paper we provide a certain *smoothness* assumption (see, e.g., [17,18]):

Assumption 2 For each $i \in \mathbb{N}_{1:n_x}$, let $\mathsf{k}_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$ be a given, continuously differentiable kernel function and $\mathcal{H}_{\mathsf{k}_i}$ be the reproducing kernel Hilbert space (RKHS) corresponding to k_i with the induced norm denoted by $\|\cdot\|_{\mathsf{k}_i}$. Then, for each $i \in \mathbb{N}_{1:n_x}$, it is assumed that $d_i \in \mathcal{H}_{\mathsf{k}_i}$. Moreover, an upper bound of the RKHS norm $\|d_i\|_{\mathsf{k}_i} \leq B_i$ is available. □

Assumption 2 implies that each $d_i : \mathbb{R}^{n_x} \to \mathbb{R}$ is characterized of the form $d_i(x) = \sum_{n=1}^{\infty} \alpha_n \mathsf{k}_i(x, x_n)$, where $x_n \in \mathbb{R}^{n_x}$, $n \in \mathbb{N}_{>0}$ are the representer points and $\alpha_n \in \mathbb{R}$, $n \in \mathbb{N}_{>0}$ are the parameters that it is necessary to decay sufficiently fast as n increases. The induced norm is given by $\|d_i\|_{\mathsf{k}_i}^2 = \sum_{n=1}^{\infty} \sum_{n'=1}^{\infty} \alpha_n \alpha_{n'} \mathsf{k}_i(x_n, x_{n'})$. In general, obtaining a large enough, yet not too conservative bound for $\|d_i\|_{\mathsf{k}_i}$ is hard; nevertheless, there exist several ways to compute an upper bound of $\|d_i\|_{\mathsf{k}_i}$ (see, e.g., [35]). The overview of how to compute an upper bound of $\|d_i\|_{\mathsf{k}_i}$ is given in Appendix A of [34], and we refer the interested reader to [35] for a more detailed discussion.

Assumption 2 allows us to show the following result:

Lemma 1 Suppose that Assumption 2 holds. Then, it follows that $|d_i(x_1) - d_i(x_2)| \leq L_i \sqrt{\|x_1 - x_2\|_\infty}$, for all $x_1, x_2 \in \mathcal{X}$, where $L_i = B_i \sqrt{2 \|\partial \mathsf{k}_i / \partial x\|_\infty}$. □

For the proof, see [34].

Remark 1 *(On selecting* k_i*)* Assumption 2 implies that the kernel function k_i should be chosen apriori. Potential candidates of this kernel function are: $\mathsf{k}_i(x, y) = e^{-a\|x-y\|^2}$, $\mathsf{k}_i(x, y) = (b + \|x - y\|^2)^{-a}$, where a, b are positive constants. A useful property of employing these kernel functions is the *universal approximation property*, i.e., the RKHSs are dense in the space of all continuous functions over any compact set (see, e.g., [36]). Hence, if the kernel function is selected as above, any continuous function can be estimated arbitrarily well by a function that lies in the RKHS. Note that there are indeed other kernels satisfying the universal approximation property, which may also be useful to be employed (see, e.g., [36]). □

3.2 Controlled invariant set and safety controller

A sequence $x_0, x_1, x_2, \ldots \in \mathbb{R}^{n_x}$ is called a *trajectory* of the system (3), if there exist $u_0, u_1, u_2, \ldots \in \mathcal{U}$, $v_0, v_1, v_2, \ldots \in \mathcal{V}$ such that $x_0 = \bar{x}$, $x_{t+1} = f(x_t, u_t) + d(x_t) + v_t$, $\forall t \in \mathbb{N}_{\geq 0}$. Moreover, a *controller* is defined as a set-valued mapping from each state onto the set of control inputs, i.e., $C : \mathbb{R}^{n_x} \to 2^{\mathcal{U}}$. Given C, a *controlled trajectory* is defined as any trajectory of the system (3), $x_0, x_1, x_2, \ldots \in \mathbb{R}^{n_x}$ with $u_t \in C(x_t)$, $\forall t \in \mathbb{N}_{\geq 0}$.

Now, denote by $\mathcal{X} \subset \mathbb{R}^{n_x}$ a *safe set*, in which the trajectory of the system (3) must stay for all times. It is assumed that $\mathcal{X}$ is compact and can be either convex or non-convex, and that $\bar{x} \in \mathcal{X}$. Based on the above, we define the notion of a controlled invariant set (see, e.g., [37]) and the safety controller as follows:

Definition 3 A set $\mathcal{X}_S \subseteq \mathcal{X}$ is called a *controlled invariant set* in $\mathcal{X}$, if there exists a controller $C_S : \mathbb{R}^{n_x} \to 2^{\mathcal{U}}$ such that the following holds: for every $x \in \mathcal{X}_S$, there exists $u \in C_S(x)$ such that for every $v \in \mathcal{V}$, $f(x, u) + d(x) + v \in \mathcal{X}_S$. The controller C_S is called a *safety controller*. □

That is, $\mathcal{X}_S$ is called a controlled invariant set if there exists a controller C_S such that every controlled trajectory induced by C_S (starting from anywhere in $\mathcal{X}_S$) stays in $\mathcal{X}_S$ for all times.

3.3 The goal of this paper and overview of the approach

The goal of this paper is to construct a *symbolic model* of the control system (3), which indicates an abstract expression of (3). In particular, due to the existence of the unknown function d, we here propose a *learning-based* approach, in which the symbolic model is constructed by learning the unknown function d from training data. Towards this end, we first provide an approach to construct a symbolic model for given training data (Section 4). The symbolic model is constructed based on the GP regression and the concept of an ε-ASR; for details, see Section 4. Based on the symbolic model, we proceed by developing an overall algorithm that aims at collecting the training data from scratch and constructing the symbolic model (Section 5). In particular, we propose a *safe exploration* algorithm, in which the trajectory of the system (3) must stay in $\mathcal{X}$ for all times while collecting the training data and constructing the symbolic model. As we will see later, this is achieved by iteratively updating the symbolic model, controlled invariant set and the safety controller after each step of the state-space exploration; for details, see Section 5.

4 Constructing symbolic models with Gaussian processes

In this section, we provide an approach to construct a symbolic model based on a given set of training data. In Section 4.1, we provide an approach to learn d_i with the GP regression as well as a useful error bound on d_i based on Assumption 2. In Section 4.2, we provide a way of how to construct a symbolic model from a given set of training data. In Section 4.3, we provide a safety controller synthesis, which will be useful to achieve the safe exploration provided in the next section.

4.1 Learning d with the GP regression

In this paper, we estimate each element of d, i.e., d_i, $i \in \mathbb{N}_{1:n_x}$ ($d = [d_1, d_2, \ldots, d_{n_x}]^\mathsf{T}$) by the GP regression

with the kernel function k_i. To this end, for each $i \in \mathbb{N}_{1:n_x}$ let $\mathcal{D}_{T,i} = \{X_T, Y_{T,i}\}$ be the set of input-output training data in order to estimate d_i, given by $X_T = [x_1, x_2, \ldots, x_T]$, $Y_{T,i} = [y_{1,i}, y_{2,i}, \ldots, y_{T,i}]^\mathsf{T}$, where $T \in \mathbb{N}_{>0}$ is the number of training data points and $y_{t,i} = x_{t+1,i} - f_i(x_t, u_t)$, $\forall t \in \mathbb{N}_{1:T}$ are the training outputs, with $x_{t,i}$ and $f_i(x_t, u_t)$ being the i-th element of x_t and $f(x_t, u_t)$, respectively. Note that we have $y_{t,i} = x_{t+1,i} - f_i(x_t, u_t) = d_i(x_t) + v_{t,i}$, where $v_{t,i}$ denotes the i-th ($i \in \mathbb{N}_{1:n_x}$) element of v_t with $|v_{t,i}| \leq \sigma_v$. Hence, $y_{t,i}$ represents the noisy output of $d_i(x_t)$ with the additive noise bounded by σ_v. As above, the realization of the additive noise sequence is uniformly bounded by σ_v, i.e., $|v_{t,i}| \leq \sigma_v$, $t \in \mathbb{N}_{\geq 0}$. When learning the unknown function, on the other hand, it is *approximated* that the additive noise is drawn independently from $\mathcal{N}(0, \sigma_v^2)$, aiming at employing the GP regression. Moreover, it is assumed for simplicity that the mean function for the GP prior is zero (i.e., $m(x) \equiv 0$ for all x in Section 2.1). Thus, the mean and the variance for the GP model of d_i with an arbitrary input $x \in \mathbb{R}^{n_x}$, denoted as $\mu_i(x; \mathcal{D}_{T,i})$ and $\sigma_i^2(x; \mathcal{D}_{T,i})$, are computed by

$$\mu_i(x; \mathcal{D}_{T,i}) = \mathsf{k}_{T,i}^{*\mathsf{T}}(x)(K_{T,i} + \sigma_v^2 I)^{-1} Y_{T,i}, \tag{5}$$

$$\sigma_i^2(x; \mathcal{D}_{T,i}) = \mathsf{k}_i(x, x) - \mathsf{k}_{T,i}^{*\mathsf{T}}(x)(K_{T,i} + \sigma_v^2 I)^{-1} \mathsf{k}_{T,i}^*(x), \tag{6}$$

where $K_{T,i}$ denote the covariance matrix for the kernel function k_i and $\mathsf{k}_{T,i}^*(x) = [\mathsf{k}_i(x, x_1), \ldots, \mathsf{k}_i(x, x_T)]^\mathsf{T}$.

Now, recall that the unknown function d_i lies in the RKHS corresponding to k_i (Assumption 2). Using this assumption, we can derive an *error bound* on d_i, representing how the GP posterior mean μ_i differs from the ground truth d_i:

Lemma 2 Suppose that Assumption 2 holds, and let $\mathcal{D}_{T,i} = \{X_T, Y_{T,i}\}$ be the training data for d_i with $X_T = [x_1, x_2, \ldots, x_T]$ and $Y_{T,i} = [y_{1,i}, y_{2,i}, \ldots, y_{T,i}]^\mathsf{T}$ for $T \in \mathbb{N}_{>0}$. Then, for all $x \in \mathbb{R}^{n_x}$ and $T \in \mathbb{N}_{>0}$, it follows that $d_i(x) \in \mathcal{Q}_i(x; \mathcal{D}_{T,i})$, where

$$\mathcal{Q}_i(x; \mathcal{D}_{T,i}) = [\mu_i(x; \mathcal{D}_{T,i}) \pm \beta_{T,i} \sigma_i(x; \mathcal{D}_{T,i})] \tag{7}$$

with $\beta_{T,i} = \sqrt{B_i^2 - Y_{T,i}^\mathsf{T}(K_{T,i} + \sigma_v^2 I)^{-1} Y_{T,i} + T}$. □

For the proof, see [34]. Lemma 2 means that $d_i(x)$ is shown to be in the interval set $\mathcal{Q}_i(x; \mathcal{D}_{T,i})$, which can be computed based on the training data for d_i.

Remark 2 Note that the previous methods of learning-based controller synthesis with the GP regression (e.g., [17,18]) make use of the probabilistic error bound characterized by the notion of an *information gain*; see Theorem 3 in [16]. For example, [17] employs the following error (or regret) bound:

$$d_i(x) \in [\mu_i(x; \mathcal{D}_{T,i}) \pm \sqrt{2\|d_i\|_{\mathsf{k}_i}^2 + 300\gamma_T \log^3(T/\delta)}\sigma_i(x; \mathcal{D}_{T,i})] \tag{8}$$

which holds for all $T \in \mathbb{N}_{\geq 0}$ with probability at least $1 - \delta$ ($0 < \delta < 1$), where γ_T, $T \in \mathbb{N}_{\geq 0}$ denote the information gain. In contrast to this bound, in this paper we provide the error bound in the *deterministic* form as in Lemma 2. Note that this bound is a direct consequence from computing the upper bound of the RKHS norm of the error $\mu_i(\cdot; \mathcal{D}_{T,i}) - d_i(\cdot)$ with respect to the kernel $\mathsf{k}_{T,i}(x, x') = \mathsf{k}_i(x, x') - \mathsf{k}_{T,i*}^\mathsf{T}(x)(K_{T,i} + \sigma_v^2 I)^{-1} \mathsf{k}_{T,i*}(x')$, which has been derived in the proof of Lemma 7.2 in [16] (in particular, see the first equation in the left column of page 3261 in [16]), and see also Appendix C in [34]. The probabilistic error bound (8) is more conservative than the deterministic one of Lemma 2 in the following sense. Note that (8) achieves a *deterministic* bound by setting $\delta \to 0$. However, setting $\delta \to 0$ in (8) implies $d_i(x) \in [-\infty, \infty]$ for all $T \in \mathbb{N}_{>0}$, which leads to an *unbounded* interval $\mathbb{R}$ (and is thus not useful for constructing a symbolic model). On the other hand, the error bound obtained in Lemma 2 is deterministic and *always bounded* (i.e., $d_i(x) \in \mathcal{Q}_i(x; \mathcal{D}_{T,i}) \subset \mathbb{R}$ holds for all $T \in \mathbb{N}_{>0}$). Such conservativeness might arise due to the fact that, in [16] the probabilistic error bound (8) has been derived as a *sufficient* condition to the deterministic one given in Lemma 2 (or Lemma 7.2 in [16]). To see this, note that the error bound obtained in Lemma 7.2 of [16] was further upper bounded by using the information gain γ_T (see the second to the third inequality in the top of the right column of page 3261 in [16]), as well as the concentration inequalities (see the second to the third inequality in the bottom of the right column of page 3261). Hence, from the above upper boundings, the probabilistic error bound has been obtained as the sufficient condition to the deterministic one given in Lemma 2. In this paper, we will make use of the deterministic error bound in Lemma 2 instead of the probabilistic one, since it allows us to derive an error bound that can get smaller as the number of the training data increases (see below for details). Such a property is useful to show that the controlled invariant set can enlarge as the number of training data increases, and, moreover, we can provide some computationally efficient algorithms for updating the symbolic models and the safety controller synthesis (see Section 5.1). □

Now, for every $T \in \mathbb{N}_{>0}$, it follows from Lemma 2 that $d_i(x) \in \mathcal{Q}_i(x; \mathcal{D}_{t,i})$ for all $t \in \mathbb{N}_{1:T}$. Thus, for every $T \in \mathbb{N}_{>0}$, we have $d_i(x) \in \bigcap_{t=1}^T \mathcal{Q}_i(x; \mathcal{D}_{t,i})$. Therefore, for every $T \in \mathbb{N}_{>0}$, we have $d_i(x) \in \mathcal{R}_i(x; \mathcal{D}_{T,i})$, where $\mathcal{R}_i(x; \mathcal{D}_{T,i}) = \bigcap_{t=1}^T \mathcal{Q}_i(x; \mathcal{D}_{t,i})$. From the definition of $\mathcal{R}_i(x; \mathcal{D}_{T,i})$, it follows that $\mathcal{R}_i(x; \mathcal{D}_{1,i}) \supseteq \mathcal{R}_i(x; \mathcal{D}_{2,i}) \supseteq \mathcal{R}_i(x; \mathcal{D}_{3,i}) \supseteq \cdots$. Let $\overline{r}_i(x; \mathcal{D}_{T,i}), \underline{r}_i(x; \mathcal{D}_{T,i}) \in \mathbb{R}$ be

given by

$$\overline{r}_i(x;\mathcal{D}_{T,i}) = \max\{r \in \mathbb{R} \mid r \in \mathcal{R}_i(x;\mathcal{D}_{T,i})\}, \quad (9)$$
$$\underline{r}_i(x;\mathcal{D}_{T,i}) = \min\{r \in \mathbb{R} \mid r \in \mathcal{R}_i(x;\mathcal{D}_{T,i})\}. \quad (10)$$

Since $\mathcal{R}_i(x;\mathcal{D}_{1,i}) \supseteq \mathcal{R}_i(x;\mathcal{D}_{2,i}) \supseteq \cdots$, it follows that $\overline{r}_i(x;\mathcal{D}_{T,i})$ and $\underline{r}_i(x;\mathcal{D}_{T,i})$ are *non-increasing* and *non-decreasing* with respect to T (for fixed x), respectively. Thus, $d_i(x) \in \mathcal{R}_i(x;\mathcal{D}_{T,i})$ implies that the error bound on d_i never grows or potentially gets smaller as the number of the training data increases. Note also that $d_i(x) \in \mathcal{R}_i(x;\mathcal{D}_{T,i})$ implies $|d_i(x) - \hat{d}_i(x;\mathcal{D}_{T,i})| \leq \Delta_i(x;\mathcal{D}_{T,i})$, where

$$\hat{d}_i(x;\mathcal{D}_{T,i}) = 0.5\left(\overline{r}_i(x;\mathcal{D}_{T,i}) + \underline{r}_i(x;\mathcal{D}_{T,i})\right), \quad (11)$$
$$\Delta_i(x;\mathcal{D}_{T,i}) = 0.5\left(\overline{r}_i(x;\mathcal{D}_{T,i}) - \underline{r}_i(x;\mathcal{D}_{T,i})\right). \quad (12)$$

4.2 Constructing symbolic models from training data

Let us now construct a symbolic model of the system (3) provided that the training data is obtained. We start by showing that the system (3) can be described within the class of a transition system (Definition 1) as follows:

Definition 4 A *transition system* induced by the system (3) is a quadruple $S = (\mathbb{R}^{n_x}, x_0, \mathcal{U}, G)$, where:

- $\mathbb{R}^{n_x}$ is a set of states;
- $x_0 \in \mathbb{R}^{n_x}$ is an initial state;
- $\mathcal{U} \subset \mathbb{R}^{n_u}$ is a set of inputs;
- $G : \mathbb{R}^{n_x} \times \mathcal{U} \to 2^{\mathbb{R}^{n_x}}$ is a transition map, where $x^+ \in G(x,u)$ iff there exists $v \in \mathcal{V}$ such that $x^+ = f(x,u) + d(x) + v$. □

Based on the transition system S, a *symbolic model* of S is constructed by discretizing the state and the input spaces, whose transitions are defined based on the training data $\mathcal{D}_{T,i}$, $i \in \mathbb{N}_{1:n_x}$. More specifically, the symbolic model is constructed with a tuple $\mathsf{q} = (\mathcal{D}_T, \eta_x, \eta_u, \varepsilon)$, where

- $\mathcal{D}_T = \{\mathcal{D}_{T,1}, \ldots, \mathcal{D}_{T,n_x}\}$ is the set of training data;
- $\eta_x \in \mathbb{R}_{>0}$ is the discretization parameter for the state space $\mathbb{R}^{n_x}$;
- $\eta_u \in \mathbb{R}_{>0}$ is the discretization parameter for the input space $\mathcal{U}$;
- $\varepsilon \in \mathbb{R}_{>0}$ is the parameter for the precision.

The corresponding symbolic model is denoted as S_q and formally defined as follows:

Definition 5 Let $S = (\mathbb{R}^{n_x}, x_0, \mathcal{U}, G)$ be the transition system induced by the system (3). Given $\mathsf{q} = (\mathcal{D}_T, \eta_x, \eta_u, \varepsilon)$, a symbolic model of S is defined as a quadruple $S_\mathsf{q} = (\mathcal{X}_\mathsf{q}, x_{\mathsf{q}0}, \mathcal{U}_\mathsf{q}, G_\mathsf{q})$, where

- $\mathcal{X}_\mathsf{q} = [\mathbb{R}^{n_x}]_{\eta_x}$ is a set of states;
- $x_{\mathsf{q}0} \in \mathcal{X}_\mathsf{q}$ is an initial state satisfying $x_{\mathsf{q}0} \in \mathsf{Nearest}_{\mathcal{X}_\mathsf{q}}(x_0)$;
- $\mathcal{U}_\mathsf{q} = [\mathcal{U}]_{\eta_u}$ is a set of inputs;
- $G_\mathsf{q} : \mathcal{X}_\mathsf{q} \times \mathcal{U}_\mathsf{q} \to 2^{\mathcal{X}_\mathsf{q}}$ is a transition map, where $x^+_\mathsf{q} \in G_\mathsf{q}(x_\mathsf{q}, u_\mathsf{q})$ iff $x^+_{\mathsf{q},i} \in [\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T,i}), \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T,i})]$, $\forall i \in \mathbb{N}_{1:n_x}$, where $x^+_{\mathsf{q},i}$ is the i-th element of x^+_q, and

$$\overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T,i}) = \overline{r}_i(x_\mathsf{q}; \mathcal{D}_{T,i}) + f_i(x_\mathsf{q}, u_\mathsf{q}) + \sigma_v + \left(L_f\varepsilon + L_i\sqrt{\varepsilon} + \eta_x\right) \quad (13)$$
$$\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T,i}) = \underline{r}_i(x_\mathsf{q}; \mathcal{D}_{T,i}) + f_i(x_\mathsf{q}, u_\mathsf{q}) - \sigma_v - \left(L_f\varepsilon + L_i\sqrt{\varepsilon} + \eta_x\right). \quad (14)$$

Recall that L_f is the Lipschitz constant for the function f, and L_i is defined in Lemma 1. Moreover, $\overline{r}_i$, $\underline{r}_i$ are defined in (9) and (10), respectively. As shown in Definition 5, the symbolic model provides an abstract expression of S, in the sense that it considers the transitions only among the *discretized* points in the state and the input spaces. The following result indeed shows that there exists an ε-ASR from S_q to S:

Proposition 1 Suppose that Assumptions 1,2 hold, and let $S = (\mathbb{R}^{n_x}, x_0, \mathcal{U}, G)$. Moreover, given $\mathsf{q} = (\mathcal{D}_T, \eta_x, \eta_u, \varepsilon)$ with $\varepsilon \geq \eta_x$, let $S_\mathsf{q} = (\mathcal{X}_\mathsf{q}, x_{\mathsf{q}0}, \mathcal{U}_\mathsf{q}, G_\mathsf{q})$ be the symbolic model of S in Definition 5. Then,

$$R(\varepsilon) = \{(x_\mathsf{q}, x) \in \mathcal{X}_\mathsf{q} \times \mathbb{R}^{n_x} \mid \|x_\mathsf{q} - x\|_\infty \leq \varepsilon\} \quad (15)$$

is an ε-ASR from S_q to S. □

The proof follows in the same way to [8] and is given in [34]. In addition to the above, we also have the following result:

Lemma 3 Let $\mathcal{D}_{T,i} = \{X_T, Y_{T,i}\}$, $i \in \mathbb{N}_{1:n_x}$ be the training data with $X_T = [x_1, x_2, \ldots, x_T]$ and $Y_{T,i} = [y_{1,i}, y_{2,i}, \ldots, y_{T,i}]^\mathsf{T}$ for all $T \in \mathbb{N}_{>0}$, and let $\mathcal{D}_T = \{\mathcal{D}_{T,1}, \ldots, \mathcal{D}_{T,n_x}\}$, $T \in \mathbb{N}_{>0}$. Moreover, for any $T_1, T_2 \in \mathbb{N}_{>0}$ with $T_1 \leq T_2$, let $\mathsf{q}_1 = (\mathcal{D}_{T_1}, \eta_x, \eta_u, \varepsilon)$ and $\mathsf{q}_2 = (\mathcal{D}_{T_2}, \eta_x, \eta_u, \varepsilon)$ and let S_{q_1} and S_{q_2} be the corresponding symbolic models according to Definition 5. Then, the relation

$$R = \{(x_\mathsf{q}, x'_\mathsf{q}) \in \mathcal{X}_\mathsf{q} \times \mathcal{X}_\mathsf{q} \mid x_\mathsf{q} = x'_\mathsf{q}\} \quad (16)$$

is a 0-ASR from S_{q_1} to S_{q_2}. □

PROOF. Let the two symbolic models be given by $S_{\mathsf{q}_1} = (\mathcal{X}_{\mathsf{q}_1}, x_{\mathsf{q}_1 0}, \mathcal{U}_{\mathsf{q}_1}, G_{\mathsf{q}_1})$, $S_{\mathsf{q}_2} = (\mathcal{X}_{\mathsf{q}_2}, x_{\mathsf{q}_2 0}, \mathcal{U}_{\mathsf{q}_2}, G_{\mathsf{q}_2})$. Note that $\mathcal{X}_{\mathsf{q}_1} = \mathcal{X}_{\mathsf{q}_2} = [\mathbb{R}^{n_x}]_{\eta_x}$, $x_{\mathsf{q}_1 0} = x_{\mathsf{q}_2 0}$ and $\mathcal{U}_{\mathsf{q}_1} = \mathcal{U}_{\mathsf{q}_2} = [\mathcal{U}]_{\eta_x}$, since we use the same discretization parameters η_x, η_u for both q_1 and q_2. Hence, the condition (C.1) in Definition 2 holds. The condition (C.2) holds from the definition of R (16). To show the condition (C.3), let us recall that for every $x_\mathsf{q} \in [\mathbb{R}^{n_x}]_{\eta_x}$,

$\overline{r}_i(x_\mathsf{q}; \mathcal{D}_{T,i})$ (resp. $\underline{r}_i(x_\mathsf{q}; \mathcal{D}_{T,i})$) is non-increasing (resp. non-decreasing) with respect to T. Hence, for every $x_\mathsf{q} \in [\mathbb{R}^{n_x}]_{\eta_x}$ and $u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$, we have

$$[\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_2,i}), \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_2,i})] \subseteq [\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_1,i}), \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_1,i})], \quad (17)$$

or in other words, $G_{\mathsf{q}_2}(x_\mathsf{q}, u_\mathsf{q}) \subseteq G_{\mathsf{q}_1}(x_\mathsf{q}, u_\mathsf{q})$. This directly means that the condition (C.3) in Definition 2 holds. Therefore, it is shown that the relation (16) is a 0-ASR from S_{q_1} to S_{q_2}.

Lemma 3 implies that, since $G_{\mathsf{q}_2}(x_\mathsf{q}, u_\mathsf{q}) \subseteq G_{\mathsf{q}_1}(x_\mathsf{q}, u_\mathsf{q})$ for every $x_\mathsf{q} \in [\mathbb{R}^{n_x}]_{\eta_x}$ and $u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$, the redundant transitions that are present in the symbolic model can be removed by increasing the number of the training data. This is due to the fact that the uncertainty (or the error bound) on the unknown function d can be smaller as the training data increases (see Section 4.1).

4.3 Synthesizing a safety controller

Given $\mathsf{q} = (\mathcal{D}_T, \eta_x, \eta_u, \varepsilon)$, suppose that the symbolic model S_q is obtained according to Definition 5. Based on the symbolic model, we can find a controlled invariant set $\mathcal{X}_S$ in $\mathcal{X}$ and the corresponding safety controller C_S by employing a *safety game*, see, e.g., [3]. The algorithm of the safety game is illustrated in Algorithm 1. In the algorithm, the operator $\mathrm{Pre}_{S_\mathsf{q}} : 2^{\mathcal{X}_\mathsf{q}} \to 2^{\mathcal{X}_\mathsf{q}}$ is called a *predecessor operator* and is defined by

$$\mathrm{Pre}_{S_\mathsf{q}}(\mathcal{Q}) = \{x_\mathsf{q} \in \mathcal{Q} \mid \exists u_\mathsf{q} \in \mathcal{U}_\mathsf{q} : G_\mathsf{q}(x_\mathsf{q}, u_\mathsf{q}) \subseteq \mathcal{Q}\}, \quad (18)$$

for a given $\mathcal{Q} \subseteq \mathcal{X}_\mathsf{q}$. That is, $\mathrm{Pre}_{S_\mathsf{q}}(\mathcal{Q})$ is the set of all states in $\mathcal{Q}$, for which there exists a control input in $\mathcal{U}_\mathsf{q}$ such that all the corresponding successors are inside $\mathcal{Q}$. The controlled invariant set $\mathcal{X}_S$ is computed based on the fixed point set of $\mathcal{Q}_\ell$ (i.e., $\mathcal{X}_{S,\mathsf{q}}$). In particular, if $\mathcal{X}_{S,\mathsf{q}}$ is non-empty, the controlled invariant set is computed based on the ε-ASR $R(\varepsilon)$ (line 9). On the other hand, if $\mathcal{X}_{S,\mathsf{q}}$ is empty, it indicates that the controlled invariant set is not found (and so we set $\mathcal{X}_S \leftarrow \varnothing$ as shown in line 13). Roughly speaking, $C_{S,\mathsf{q}}$ (line 10) serves as a safety controller for the symbolic model S_q, and the safety controller C_S for S is *refined* based on the ε-ASR $R(\varepsilon)$ (line 11). Note that Algorithm 1 is guaranteed to terminate after a finite number of iteration, since $[\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}$ and $[\mathcal{U}]_{\eta_u}$ are both finite. The following result is an immediate consequence from the fact that $R(\varepsilon)$ is the ε-ASR from S_q to S and thus the proof is omitted (see, e.g., [3]).

Lemma 4 Suppose that for given S_q and $\mathcal{X}$, Algorithm 1 is implemented and $\mathcal{X}_S \neq \varnothing$. Then, $\mathcal{X}_S$ is a controlled invariant set in $\mathcal{X}$, and C_S is the corresponding safety controller. □

Algorithm 1 $\mathsf{SafeCon}(S_\mathsf{q}, \mathcal{X})$ (safety controller synthesis).

Input: S_q (symbolic model of S), $\mathcal{X}$ (safe set);
Output: $\mathcal{X}_S$ (if $\varnothing \neq \mathcal{X}_S$, it yields a controlled invariant set in $\mathcal{X}$), C_S (if $\varnothing \neq \mathcal{X}_S$, it yields a safety controller);
1: $\ell \leftarrow 0$;
2: $\mathcal{Q}_\ell \leftarrow [\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}$;
3: **repeat**
4: $\quad \ell \leftarrow \ell + 1$;
5: $\quad \mathcal{Q}_\ell \leftarrow \mathrm{Pre}_{S_\mathsf{q}}(\mathcal{Q}_{\ell-1})$;
6: **until** $\mathcal{Q}_{\ell-1} = \mathcal{Q}_\ell$
7: $\mathcal{X}_{S,\mathsf{q}} \leftarrow \mathcal{Q}_\ell$;
8: **if** $\varnothing \neq \mathcal{X}_{S,\mathsf{q}}$ **then**
9: $\quad \mathcal{X}_S \leftarrow \{x \in \mathcal{X} | \exists x_\mathsf{q} \in \mathcal{X}_{S,\mathsf{q}}, (x_\mathsf{q}, x) \in R(\varepsilon)\}$;
10: $\quad C_{S,\mathsf{q}}(x_\mathsf{q}) \leftarrow \{u_\mathsf{q} \in \mathcal{U}_\mathsf{q} | G_\mathsf{q}(x_\mathsf{q}, u_\mathsf{q}) \subseteq \mathcal{X}_{S,\mathsf{q}}\}, \forall x_\mathsf{q} \in \mathcal{X}_{S,\mathsf{q}}$;
11: $\quad C_S(x) \leftarrow \{C_{S,\mathsf{q}}(x_\mathsf{q}) | (x_\mathsf{q}, x) \in R(\varepsilon)\}, \forall x \in \mathcal{X}_S$;
12: **else**
13: $\quad \mathcal{X}_S \leftarrow \varnothing$, $C_S(x) \leftarrow \varnothing$, $\forall x \in \mathcal{X}$ (which indicates that the controlled invariant set and safety controller are not found);
14: **end if**

In addition to the above, we also have the following result:

Lemma 5 Let $\mathcal{D}_{T,i} = \{X_T, Y_{T,i}\}$, $i \in \mathbb{N}_{1:n_x}$ be the training data with $X_T = [x_1, x_2, \ldots, x_T]$ and $Y_{T,i} = [y_{1,i}, y_{2,i}, \ldots, y_{T,i}]^\mathsf{T}$ for all $T \in \mathbb{N}_{>0}$, and let $\mathcal{D}_T = \{\mathcal{D}_{T,1}, \ldots, \mathcal{D}_{T,n_x}\}$, $T \in \mathbb{N}_{>0}$. Moreover, for any $T_1, T_2 \in \mathbb{N}_{>0}$ with $T_1 \leq T_2$, let $\mathsf{q}_1 = (\mathcal{D}_{T_1}, \eta_x, \eta_u, \varepsilon)$ and $\mathsf{q}_2 = (\mathcal{D}_{T_2}, \eta_x, \eta_u, \varepsilon)$ and let S_{q_1} and S_{q_2} be the corresponding symbolic models according to Definition 5. In addition, let $\mathcal{X}_{S_1}, \mathcal{X}_{S_2}$ be the resulting controlled invariant sets by executing $\mathsf{SafeCon}(S_{\mathsf{q}_1}, \mathcal{X})$ and $\mathsf{SafeCon}(S_{\mathsf{q}_2}, \mathcal{X})$, respectively. Then, $\mathcal{X}_{S_1} \subseteq \mathcal{X}_{S_2}$. □

In essence, Lemma 5 means that the controlled invariant set does not shrink or can be enlarged by increasing the number of training data. As previously mentioned, this is due to that the symbolic model becomes more and more accurate (i.e., the redundant transitions are removed) as the training data increases, since the error bound on d can be smaller as the training data increases. While Lemma 5 might trivially follow from the existence of a 0-ASR from S_{q_1} to S_{q_2} (see Lemma 3), we here provide a detailed proof below, since the proof procedure will be useful to provide an approach to reduce the computational load for the safety controller synthesis (for details, see Section 5.1).

PROOF. Let the two symbolic models be given by $S_{\mathsf{q}_1} = (\mathcal{X}_{\mathsf{q}_1}, x_{\mathsf{q}_1 0}, \mathcal{U}_{\mathsf{q}_1}, G_{\mathsf{q}_1})$, $S_{\mathsf{q}_2} = (\mathcal{X}_{\mathsf{q}_2}, x_{\mathsf{q}_2 0}, \mathcal{U}_{\mathsf{q}_2}, G_{\mathsf{q}_2})$. Then, from the proof of Lemma 3, it follows that $G_{\mathsf{q}_2}(x_\mathsf{q}, u_\mathsf{q}) \subseteq G_{\mathsf{q}_1}(x_\mathsf{q}, u_\mathsf{q})$ for every $x_\mathsf{q} \in [\mathbb{R}^{n_x}]_{\eta_x}$ and $u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$. Now, let $\mathcal{Q}_{1,\ell}$, $\mathcal{Q}_{2,\ell}$, $\ell = 0, 1, \ldots$ be the sets of $\mathcal{Q}_\ell$ obtained by executing $\mathsf{SafeCon}(S_{\mathsf{q}_1}, \mathcal{X})$ and $\mathsf{SafeCon}(S_{\mathsf{q}_2}, \mathcal{X})$, respectively. Note that $\mathcal{Q}_{1,0} = \mathcal{Q}_{2,0}$.

Algorithm 2 Learning-based symbolic abstractions with safe exploration (overall, main algorithm).

Input: x_0 (initial state), $C_{S,\text{init}}$ (initial safety controller), $\eta_x, \eta_u, \varepsilon$ (some parameters for the symbolic model), $T_{\text{exp}} \in \mathbb{N}_{>0}$ (number of training data collected for each iteration of safe exploration);
Output: S_{q_N} (symbolic model);
1: $\mathcal{D}_{0,i} \leftarrow \varnothing$, $\forall i \in \mathbb{N}_{1:n_x}$;
2: $\mathcal{D}_0 \leftarrow \{\mathcal{D}_{0,1}, \ldots, \mathcal{D}_{0,n_x}\}$;
3: $T_1 \leftarrow T_{\text{exp}}$;
4: $\{x_{T_1}, \mathcal{D}_{T_1}\} \leftarrow \mathsf{SafeExp}(x_0, T_{\text{exp}}, C_{S,\text{init}}, \mathcal{D}_0)$;
5: $\mathsf{q}_1 \leftarrow \{\mathcal{D}_{T_1}, \eta_x, \eta_u, \varepsilon\}$;
6: $S_{\mathsf{q}_1} \leftarrow (\mathcal{X}_{\mathsf{q}_1}, x_{\mathsf{q}_1 0}, \mathcal{U}_{\mathsf{q}_1}, G_{\mathsf{q}_1})$ (Definition 5);
7: $\{\mathcal{X}_{S,1}, C_{S,1}\} \leftarrow \mathsf{SafeCon}(S_{\mathsf{q}_1}, \mathcal{X})$ (Algorithm 1);
8: **if** $\varnothing \neq \mathcal{X}_{S,1}$ **then**
9: **repeat**
10: $T_{N+1} \leftarrow T_N + T_{\text{exp}}$;
11: $\{x_{T_{N+1}}, \mathcal{D}_{T_{N+1}}\} \leftarrow \mathsf{SafeExp}(x_{T_N}, T_{\text{exp}}, C_{S,N}, \mathcal{D}_{T_N})$
12: $N \leftarrow N + 1$;
13: $\mathsf{q}_N \leftarrow \{\mathcal{D}_{T_N}, \eta_x, \eta_u, \varepsilon\}$;
14: $S_{\mathsf{q}_N} \leftarrow (\mathcal{X}_{\mathsf{q}_N}, x_{\mathsf{q}_N 0}, \mathcal{U}_{\mathsf{q}_N}, G_{\mathsf{q}_N})$ (Definition 5);
15: $\{\mathcal{X}_{S,N}, C_{S,N}\} \leftarrow \mathsf{SafeCon}(S_{\mathsf{q}_N}, \mathcal{X})$ (Algorithm 1);
16: **until** $\mathcal{X}_{S,N-1} = \mathcal{X}_{S,N}$
17: **end if**

Hence, it follows that

$$G_{\mathsf{q}_1}(x_{\mathsf{q}}, u_{\mathsf{q}}) \subseteq \mathcal{Q}_{1,0} \implies G_{\mathsf{q}_2}(x_{\mathsf{q}}, u_{\mathsf{q}}) \subseteq \mathcal{Q}_{2,0}. \quad (19)$$

Thus, we obtain $\text{Pre}_{S_{\mathsf{q}_1}}(\mathcal{Q}_{1,0}) \subseteq \text{Pre}_{S_{\mathsf{q}_2}}(\mathcal{Q}_{2,0})$. Hence, from the fact that $\mathcal{Q}_{1,0} = \mathcal{Q}_{2,0}$ and line 5 in Algorithm 1, it follows that $\mathcal{Q}_{1,1} \subseteq \mathcal{Q}_{2,1}$. By recursively applying the same reasoning as above, it then follows that $\mathcal{Q}_{1,\ell} \subseteq \mathcal{Q}_{2,\ell}$, $\ell = 0, 1, \ldots$. In other words, we have $\mathcal{X}_{S_1,\mathsf{q}} \subseteq \mathcal{X}_{S_2,\mathsf{q}}$ and namely, $\mathcal{X}_{S_1} \subseteq \mathcal{X}_{S_2}$.

5 Learning-based safe symbolic abstractions

In this section we present an overall algorithm that aims at collecting the training data from scratch and constructing the symbolic model while achieving the safe exploration. Before providing the algorithm, we need to make the following assumption:

Assumption 3 There exists a *known* safety controller $C_{S,\text{init}} : \mathcal{X} \to 2^{\mathcal{U}}$ such that any trajectory induced by $C_{S,\text{init}}$ stays in the safety set $\mathcal{X}$ for all times. □

Assumption 3 implies the existence of an initial safety controller, so that the training data can be collected at the initial phase. The initial safety controller $C_{S,\text{init}}$ may be obtained by employing an expert or heuristically based on the nominal model $f(x_k, u_k)$; for details, see Remark 5 in [34].

The overall learning algorithm is shown in Algorithm 2 and the details are described as follows. The algorithm starts by initializing the training data by applying the initial safety controller and then updating the controlled invariant set and the safety controller (line 1–line 7). Then, if $\varnothing \neq \mathcal{X}_{S,1}$ (i.e., $\mathcal{X}_{S,1}$ is a controlled invariant set in $\mathcal{X}$), we move on to the iteration (lines 9–16). In the iteration, we first update T_{N+1} (line 10). Roughly speaking, T_N, $N \in \mathbb{N}_{\geq 0}$ indicate the number of training data that has been collected until the N-th iteration of Algorithm 2. The algorithm proceeds by executing a *safe exploration* algorithm $\mathsf{SafeExp}$ (line 11), which aims at collecting the new training data while guaranteeing safety. In detail, the safe exploration algorithm is shown in Algorithm 3. In the algorithm, the control input u_t (line 4) is computed as follows:

$$u_t = \begin{cases} \text{select arbitrarily from } C_{S,N}(x_t), (\text{if } N = 0), & (21) \\ \underset{u \in C_{S,N}(x_t)}{\arg\max} \sum_{i=1}^{n_x} \sigma_i^2(\hat{x}_i^+; \mathcal{D}_{T_N,i}), \quad (\text{if } N > 0), & (22) \end{cases}$$

where $\hat{x}^+ = f(x_t, u) + \hat{d}(x_t; \mathcal{D}_{T_N})$ with $\hat{d}(x_t; \mathcal{D}_{T_N}) = [\hat{d}_1(x_t; \mathcal{D}_{T_N,1}), ..., \hat{d}_{n_x}(x_t; \mathcal{D}_{T_N,n_x})]^\mathsf{T}$. Recall that σ_i^2 and $\hat{d}_i$ are defined in (6) and (11), respectively. That is, we select the control input randomly from $C_{S,N}$ for the initial exploration, and, otherwise, select from $C_{S,N}$ such that the corresponding (predictive) next state has the largest variance on d. By doing so, the system actively explores the state-space so as to reduce the uncertainty on d and enlarge the controlled invariant set while guaranteeing safety. The exploration is given until it collects the new T_{exp} training data, and it outputs the new training data $\mathcal{D}_{T_{N+1}}$ and the current state $x_{T_{N+1}}$ after the exploration. Afterwards, the symbolic model S_{q_N} is updated with the new training data according to Definition 5 (line 13, line 14 in Algorithm 2), and the controlled invariant set $\mathcal{X}_{S,N}$ and the safety controller $C_{S,N}$ are updated by $\mathsf{SafeCon}$ (line 15 in Algorithm 2). The above procedure is

Algorithm 3 $\mathsf{SafeExp}(x_{T_N}, T_{\text{exp}}, C_{S,N}, \mathcal{D}_{T_N})$ (safe exploration).

Input: x_{T_N}(current state), T_{exp} (number of training data collected for each iteration of safe exploration), $C_{S,N}$ (safety controller),
$\mathcal{D}_{T_N}$ (current training data);
Output: $x_{T_{N+1}}$,$\mathcal{D}_{T_{N+1}}$ (updated current state and training data after the exploration);
1: $X \leftarrow \varnothing$;
2: $Y_i \leftarrow \varnothing$, $i \in \mathbb{N}_{1:n_x}$(initialize the new training data);
3: **for** $t = T_N : T_N + T_{\text{exp}} - 1$ **do**
4: Compute $u_t \in C_{S,N}(x_t)$ by (21),(22);
5: Apply u_t and measure the next state: $x_{t+1} = [x_{t+1,1}, \ldots, x_{t+1,n_x}]^\mathsf{T}$;
6: For all $i \in \mathbb{N}_{1:n_x}$, set the training data as follows:

$$X \leftarrow [X,\ x_t],\ Y_i \leftarrow [Y_i,\ x_{t+1,i} - f_i(x_t, u_t)]; \quad (20)$$

7: **end for**
8: $T_{N+1} \leftarrow T_N + T_{\text{exp}}$;
9: $\mathcal{D}_{T_{N+1},i} \leftarrow \mathcal{D}_{T_N,i} \cup \{X, Y_i\}$, $i \in \mathbb{N}_{1:n_x}$;
10: $\mathcal{D}_{T_{N+1}} \leftarrow \{\mathcal{D}_{T_{N+1},1}, \ldots, \mathcal{D}_{T_{N+1},n_x}\}$;

iterated until the controlled invariant set converges, i.e., $\mathcal{X}_{S,N-1} = \mathcal{X}_{S,N}$ (see line 12 in Algorithm 2). Note that $\mathcal{X}_{S,N}$ is computed by refining the set of discretized states, i.e., $\mathcal{X}_{S,N} = \{x \in \mathcal{X} \mid \exists x_{\mathsf{q}} \in \mathcal{X}_{S,\mathsf{q}_N}, (x_{\mathsf{q}}, x) \in R(\varepsilon)\}$ (see line 8 in Algorithm 1). Hence, we have $\mathcal{X}_{S,N-1} = \mathcal{X}_{S,N}$ if and only if $\mathcal{X}_{S,\mathsf{q}_N} = \mathcal{X}_{S,\mathsf{q}_{N-1}}$. Since $\mathcal{X}_{S,\mathsf{q}_{N-1}}$ and $\mathcal{X}_{S,\mathsf{q}_N}$ are both finite and $\mathcal{X}_{S,\mathsf{q}_{N-1}} \subseteq \mathcal{X}_{S,\mathsf{q}_N}$ (for details, see the proof of Theorem 1 below), the condition $\mathcal{X}_{S,\mathsf{q}_N} = \mathcal{X}_{S,\mathsf{q}_{N-1}}$ can be checked in a finite time (i.e., check if every $x_{\mathsf{q}} \in \mathcal{X}_{S,\mathsf{q}_N}$ is contained in $\mathcal{X}_{S,\mathsf{q}_{N-1}}$).

Regarding the overall algorithm, we can conclude the following result:

Theorem 1 Suppose that Assumptions 1–3 hold and Algorithm 2 is implemented. Then, Algorithm 2 terminates after a finite number of iteration. Moreover, the relation $R(\varepsilon) = \{(x_{\mathsf{q}}, x) \in \mathcal{X}_{\mathsf{q}} \times \mathbb{R}^{n_x} \mid \|x_{\mathsf{q}} - x\|_\infty \leq \varepsilon\}$ is an ε-ASR from S_{q_N} to S for all $N \in \mathbb{N}_{>0}$ until Algorithm 2 terminates. In addition, the *safe exploration* is achieved, i.e., during the implementation of Algorithm 3, it is shown that the trajectory of the system (3) stays in the safe set $\mathcal{X}$ for all times. □

PROOF. Let us first show that Algorithm 2 terminates after a finite number of iteration. Given $N \in \mathbb{N}_{>0}$, let $\mathcal{X}_{S,\mathsf{q}_N}$ be the set of $\mathcal{X}_{S,\mathsf{q}}$ in Algorithm 1 (line 7) computed by executing $\mathsf{SafeCon}(S_{\mathsf{q}_N}, \mathcal{X})$. Since $T_{N+1} \geq T_N$, and from the proof of Lemma 5, we obtain $\mathcal{X}_{S,\mathsf{q}_N} \subseteq \mathcal{X}_{S,\mathsf{q}_{N+1}}$. In general, it follows that $\mathcal{X}_{S,\mathsf{q}_0} \subseteq \mathcal{X}_{S,\mathsf{q}_1} \subseteq \mathcal{X}_{S,\mathsf{q}_2} \subseteq \cdots$. Note that $\mathcal{X}_{S,\mathsf{q}_N} \subseteq [\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}$ for all $N \in \mathbb{N}_{>0}$ and that $[\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}$ is finite. Hence, there exists an $N' \in \mathbb{N}_{>0}$ such that $\mathcal{X}_{S,\mathsf{q}_{N'}} = \mathcal{X}_{S,\mathsf{q}_{N'+1}}$. This in turn implies that $\mathcal{X}_{S,N'} = \mathcal{X}_{S,N'+1}$, and, therefore, Algorithm 2 terminates after a finite number of iteration. The fact that $R(\varepsilon)$ is an ε-ASR from S_{q_N} to S for all $N \in \mathbb{N}_{>0}$ (until Algorithm 2 terminates) trivially holds from Proposition 1. Moreover, we can achieve the safe exploration, since control inputs are always chosen from the safety controller $C_{S,N}$.

Note that, in order to make the implementation of Algorithm 2 tractable, it is only necessary to construct the symbolic model within the state-space $[\mathcal{X}]_{\eta_x}$. Specifically, the update of the symbolic model (line 14 in Algorithm 2) is replaced by defining a new symbolic model S_{D,q_N}:

$$S_{D,\mathsf{q}_N} \leftarrow (\mathcal{X}_{D,\mathsf{q}_N}, x_{D,\mathsf{q}_N 0}, \mathcal{U}_{D,\mathsf{q}_N}, G_{D,\mathsf{q}_N}), \tag{23}$$

where $\mathcal{X}_{D,\mathsf{q}_N} = [\mathcal{X}]_{\eta_x}$, $x_{D,\mathsf{q}_N 0} = x_{\mathsf{q}_N 0}$, $\mathcal{U}_{D,\mathsf{q}_N} = \mathcal{U}_{\mathsf{q}_N}$, and $G_{D,\mathsf{q}_N} : \mathcal{X}_{D,\mathsf{q}_N} \times \mathcal{U}_{D,\mathsf{q}_N} \to 2^{\mathcal{X}_{D,\mathsf{q}_N}}$, with $x^+_{\mathsf{q}} \in G_{D,\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}})$ if and only if $x^+_{\mathsf{q}} \in G_{\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}})$ and $G_{\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}}) \subseteq [\mathcal{X}]_{\eta_x}$ (i.e., if $G_{D,\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}}) \not\subseteq [\mathcal{X}]_{\eta_x}$ then $u_{\mathsf{q}} \notin \mathcal{U}_{D,\mathsf{q}_N}(x_{\mathsf{q}})$). It can be easily shown that the relation $R = \{(x_{\mathsf{q}}, x'_{\mathsf{q}}) \in [\mathcal{X}]_{\eta_x} \times [\mathbb{R}^{n_x}]_{\eta_x} \mid x_{\mathsf{q}} = x'_{\mathsf{q}}\}$ is a 0-ASR from S_{D,q_N} to S_{q_N}. From this and the fact that the relation $R(\varepsilon) = \{(x_{\mathsf{q}}, x) \in \mathcal{X}_{\mathsf{q}} \times \mathbb{R}^{n_x} \mid \|x_{\mathsf{q}} - x\|_\infty \leq \varepsilon\}$ is the ε-ASR from S_{q_N} to S, it is shown that the relation $R_D(\varepsilon) = \{(x_{\mathsf{q}}, x) \in [\mathcal{X}]_{\eta_x} \times \mathbb{R}^{n_x} \mid \|x_{\mathsf{q}} - x\|_\infty \leq \varepsilon\}$ is an ε-ASR from S_{D,q_N} to S (see, e.g., [8] for a detailed discussion). Hence, any controller synthesized for the symbolic model S_{D,q_N} can be refined to a controller for the original system S satisfying the same specification.

5.1 Some approaches to efficient computation

Since the symbolic model needs to be updated for *every* N, the whole re-computation of this abstraction (as well as the safety controller synthesis) for every iteration clearly leads to a heavy computational load. Therefore, in this section we provide some techniques to reduce the computational load so as to make our approach more practical. Specifically, we propose the following two approaches to speed up the abstraction and controller synthesis procedures:

- *(Lazy abstraction):* It should be expected that, the transitions are necessary to be updated only for the states where the uncertainty (or the variance) on d is sufficiently reduced by collecting the new training data. Hence, we propose a *lazy abstraction* scheme, in which, starting from the initial abstraction, transitions from states in $[\mathcal{X}]_{\eta_x}$ are then updated *only when* the reduction of the variance on d is large enough. The update of the transitions allows to reduce the redundant transitions and hence and the abstraction becomes less conservative. In the proposed procedure, we do not have to recompute the abstraction for the whole states in $[\mathcal{X}]_{\eta_x}$, but only for the states on which new training data is collected.
- *(Speeding up the computation of predecessors):* It should be expected that the main source of the heavy computation for the safety controller synthesis is the predecessor operator $\mathrm{Pre}_{S_{\mathsf{q}}}(\mathcal{Q}_\ell)$ (see (18)); clearly, checking for *every* state in $\mathcal{Q}_\ell$ if there exists a control input such that all the corresponding successors are in $\mathcal{Q}_\ell$ requires a heavy computation, as this operation needs to be done for every ℓ and N. Therefore, we propose an approach to reduce the computational load of computing this predecessor operator, in order to speed up the safety controller synthesis. In particular, we eliminate redundant computations of the predecessor operator by making use of the earlier computed predecessors.

Regarding the first approach in the above, the update of the symbolic model (line 14 in Algorithm 2) is replaced by defining a new symbolic model $\widetilde{S}_{D,\mathsf{q}_N}$:

$$\widetilde{S}_{D,\mathsf{q}_N} \leftarrow (\mathcal{X}_{D,\mathsf{q}_N}, x_{D,\mathsf{q}_N 0}, \mathcal{U}_{D,\mathsf{q}_N}, \widetilde{G}_{D,\mathsf{q}_N}), \tag{24}$$

for all $N \in \mathbb{N}_{\geq 1}$, where $\widetilde{G}_{D,\mathsf{q}_N}$ is the transition map that is (newly) constructed by applying Algorithm 4. The

Algorithm 4 Derivation of $\widetilde{G}_{D,\mathsf{q}_N}$ for all $N \in \mathbb{N}_{\geq 1}$ (lazy abstraction).

Input: $\mathcal{D}_{T_{1:N}}$(training data), $\rho \in \mathbb{R}_{>0}$ (threshold to update transitions in $\widetilde{G}_{D,\mathsf{q}_N}$), $\widetilde{G}_{D,\mathsf{q}_{N-1}}$(transition map of $\widetilde{S}_{D,\mathsf{q}_{N-1}}$ (if $N > 1$));
Output: $\widetilde{G}_{D,\mathsf{q}_N}$ (transition map of $\widetilde{S}_{D,\mathsf{q}_N}$);
1: **if** $N = 1$ (initial execution of Algorithm 4) **then**
2: $\quad \mathcal{X}^c_{\mathsf{q}} \leftarrow \varnothing$;
3: **end if**
4: **for** each $x_{\mathsf{q}} \in \mathcal{X}^c_{\mathsf{q}}$ **do**
5: $\quad$ **for** each $u_{\mathsf{q}} \in [\mathcal{U}]_{\eta_u}$ **do**
6: $\quad\quad \widetilde{G}_{D,\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}}) \leftarrow \widetilde{G}_{D,\mathsf{q}_{N-1}}(x_{\mathsf{q}}, u_{\mathsf{q}})$;
7: $\quad$ **end for**
8: **end for**
9: **for** each $x_{\mathsf{q}} \in [\mathcal{X}]_{\eta_x} \backslash \mathcal{X}^c_{\mathsf{q}}$ **do**
10: $\quad$ **for** each $u_{\mathsf{q}} \in [\mathcal{U}]_{\eta_u}$ **do**
11: $\quad\quad \mathcal{X}^+_{\mathsf{q}} \leftarrow \{x^+_{\mathsf{q}} \in [\mathbb{R}^{n_x}]_{\eta_x} \mid x^+_{\mathsf{q},i} \in [\underline{h}_i(x_{\mathsf{q}}, u_{\mathsf{q}}; \mathcal{D}_{T_N,i}), \overline{h}_i(x_{\mathsf{q}}, u_{\mathsf{q}}; \mathcal{D}_{T_N,i})], \forall i \in \mathbb{N}_{1:n_x}\}$;
12: $\quad\quad$ **if** $\mathcal{X}^+_{\mathsf{q}} \subseteq [\mathcal{X}]_{\eta_x}$ **then**
13: $\quad\quad\quad \widetilde{G}_{D,\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}}) \leftarrow \mathcal{X}^+_{\mathsf{q}}$;
14: $\quad\quad\quad$ **if** $\Delta_i(x_{\mathsf{q}}; \mathcal{D}_{T_N,i}) < \rho$ for all $i \in \mathbb{N}_{1:n_x}$ **then**
15: $\quad\quad\quad\quad \mathcal{X}^c_{\mathsf{q}} \leftarrow \mathcal{X}^c_{\mathsf{q}} \cup \{x_{\mathsf{q}}\}$;
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: **end for**

core element of Algorithm 4 is the set $\mathcal{X}^c_{\mathsf{q}}$. This set is defined as the empty set at the initial execution of Algorithm 4 ($N = 1$), i.e., $\mathcal{X}^c_{\mathsf{q}} = \varnothing$ and then it is updated for $N > 1$ (as detailed below). Note that since $\mathcal{X}^c_{\mathsf{q}} = \varnothing$ for the initial execution, the procedure of line 4–line 8 is not implemented for $N = 1$. As shown in line 9–line 19, for each state in $[\mathcal{X}]_{\eta_x} \backslash \mathcal{X}^c_{\mathsf{q}}$ and each input in $[\mathcal{U}]_{\eta_u}$, the corresponding transition map $\widetilde{G}_{D,\mathsf{q}_N}(x_{\mathsf{q}}, u_{\mathsf{q}})$ is updated (see lines 11 and 13). More importantly, as shown in lines 14 and 15, if $\Delta_i(x_{\mathsf{q}}; \mathcal{D}_{T_N,i}) < \rho$ holds for all $i \in \mathbb{N}_{1:n_x}$, then x_{q} is *added* to $\mathcal{X}^c_{\mathsf{q}}$ (ρ denotes a user-defined threshold). Recall that $\Delta_i(x_{\mathsf{q}}; \mathcal{D}_{T_N,i})$ represents the length of the confidence interval, or *uncertainty* for $d_i(x_{\mathsf{q}})$ given the training data $\mathcal{D}_{T_N,i}$ (see (12)). Hence, if the uncertainty of the state x_{q} becomes small enough, then x_{q} is added to $\mathcal{X}^c_{\mathsf{q}}$. As shown in line 4 to 8, if x_{q} is added to $\mathcal{X}^c_{\mathsf{q}}$, the transitions from x_{q} is kept the same as the previous iteration afterwards (see line 6). That is, the transitions from x_{q} are no more updated once the corresponding uncertainty becomes small enough. This is reasonable because the states having small uncertainties will not have redundant transitions and so it is no longer necessary to update the transition map. Moreover, this will indeed speed up the construction of the transition map, since the transitions from some of the states are not necessary to be updated once their uncertainties become small.

In summary, the symbolic model is given by (24), where the corresponding transition map $\widetilde{G}_{D,\mathsf{q}_N}$ is computed by executing Algorithm 4 for all $N \in \mathbb{N}_{\geq 1}$ (until Algorithm 2 terminates). The computational complexity of Algorithm 4 is provided as follows. For the initial execution of Algorithm 4 ($N = 1$), we have $\mathcal{X}^c_{\mathsf{q}} = \varnothing$ and thus the transition map G_{D,q_N} is computed for all states in $[\mathcal{X}]_{\eta_x}$ and all inputs in $[\mathcal{U}]_{\eta_u}$. Hence, the computational complexity of constructing the transition map is $\mathcal{O}(|[\mathcal{X}]_{\eta_x}|\,|[\mathcal{U}]_{\eta_u}|c(T_N))$, where $c(T_N)$ denotes the computational complexity of the one-step reachable states (line 11 in Algorithm 4). Here, the computational complexity of the one-step reachable states depends on the data size T_N, since the computations of $\underline{h}_i(x_{\mathsf{q}}, u_{\mathsf{q}}; \mathcal{D}_{T_N,i})$ and $\overline{h}_i(x_{\mathsf{q}}, u_{\mathsf{q}}; \mathcal{D}_{T_N,i})$ involve the computations of the GP mean and variance. For example, standard computation of the GP mean/variance requires a cubic complexity $\mathcal{O}(T_N^3)$ due to the inversion of the $T_N \times T_N$ matrix. Note that the construction of the symbolic model for the *known* dynamics requires $\mathcal{O}(|[\mathcal{X}]_{\eta_x}|\,|[\mathcal{U}]_{\eta_u}|)$, because we need to define the transition maps for every pair of the state and the control input $(x_{\mathsf{q}}, u_{\mathsf{q}}) \in [\mathcal{X}]_{\eta_x} \times [\mathcal{U}]_{\eta_u}$. Hence, the computational complexity of our approach additionally requires the multiplication of $c(T_N)$ (in contrast to the one of the abstraction scheme with the known dynamics). This is clear because the dynamics is here estimated by a non-parametric (or, GP) model based on training data. Now, consider $N > 1$. As shown in Algorithm 4, if $x_{\mathsf{q}} \in [\mathcal{X}]_{\eta_x} \backslash \mathcal{X}^c_{\mathsf{q}}$, transitions are re-computed for all $u_{\mathsf{q}} \in [\mathcal{U}]_{\eta_u}$ (line 9–line 19), and otherwise, transitions from x_{q} are directly set as the previous ones of $N-1$ (line 4–line 8). Hence, the computational complexity of Algorithm 4 is

$$\mathcal{O}\Big(\underbrace{(|[\mathcal{X}]_{\eta_x}| - |\mathcal{X}^c_{\mathsf{q}}|) \cdot |[\mathcal{U}]_{\eta_u}| \cdot c(T_N)}_{line\,9-line\,19} + \underbrace{|\mathcal{X}^c_{\mathsf{q}}| \cdot |[\mathcal{U}]_{\eta_u}|}_{line\,4-line\,8}\Big)$$
$$= \mathcal{O}\Big(|[\mathcal{X}]_{\eta_x}| \cdot |[\mathcal{U}]_{\eta_u}| \cdot c(T_N) - |\mathcal{X}^c_{\mathsf{q}}| \cdot |[\mathcal{U}]_{\eta_u}|\big(c(T_N) - 1\big)\Big).$$

This implies that Algorithm 4 becomes faster as the cardinality of $\mathcal{X}^c_{\mathsf{q}}$ becomes larger, i.e., the number of states having small uncertainties is larger. Therefore, it is expected that the execution time of Algorithm 4 will be shorter as the state-space exploration progresses and the uncertainty on the unknown function d becomes smaller.

The following result shows that the existence of an ε-ASR is still guaranteed from $\widetilde{S}_{D,\mathsf{q}_N}$ to S.

Theorem 2 Suppose that Assumptions 1–3 hold and Algorithm 2 is implemented, in which the symbolic model is given by (24) whose transition map $\widetilde{G}_{D,\mathsf{q}_N}$ is computed by executing Algorithm 4 for all $N \in \mathbb{N}_{\geq 1}$. Then, for every $N \in \mathbb{N}_{>0}$, the relation $R_D(\varepsilon) = \{(x_{\mathsf{q}}, x) \in [\mathcal{X}]_{\eta_x} \times \mathbb{R}^{n_x} \mid \|x_{\mathsf{q}} - x\|_\infty \leq \varepsilon\}$ is an ε-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to S. □

PROOF. The result follows by induction. For $N = 1$, R_D is the ε-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to S, since $\widetilde{S}_{D,\mathsf{q}_N} = S_{D,\mathsf{q}_N}$

and R_D is the ε-ASR from S_{D,q_N} to S (see the discussion after Theorem 1). For a given $N \in \mathbb{N}_{>1}$, assume that R_D is the ε-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to S, and suppose that, at the next iteration $N+1$, $\widetilde{G}_{D,\mathsf{q}_{N+1}}$ is given by Algorithm 4. In what follows, it is shown that there exists a 0-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to $\widetilde{S}_{D,\mathsf{q}_{N+1}}$. From the derivation of $\widetilde{G}_{D,\mathsf{q}_{N+1}}$ in Algorithm 4, for every $x_\mathsf{q} \in [\mathcal{X}]_{\eta_x}, u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$ with $\widetilde{G}_{D,\mathsf{q}_N}(x_\mathsf{q}, u_\mathsf{q}) \neq \varnothing$, it follows either $\widetilde{G}_{D,\mathsf{q}_{N+1}}(x_\mathsf{q}, u_\mathsf{q}) = \widetilde{G}_{D,\mathsf{q}_N}(x_\mathsf{q}, u_\mathsf{q})$, or $\widetilde{G}_{D,\mathsf{q}_{N+1}}(x_\mathsf{q},u_\mathsf{q}) = \{x^+_\mathsf{q} \in [\mathcal{X}]_{\eta_x} \,|x^+_{\mathsf{q},i} \in [\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N+1},i}),\ \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N+1},i})], \forall i \in \mathbb{N}_{1:n_x}\}$. Note that for the latter case, we have

$$[\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N+1},i}), \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N+1},i})] \subseteq [\underline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N'},i}), \overline{h}_i(x_\mathsf{q}, u_\mathsf{q}; \mathcal{D}_{T_{N'},i})],$$

for all $N' \leq N$, since $T_{N'} \leq T_N$ (see the proof of Lemma 3). Hence, for every $x_\mathsf{q} \in [\mathcal{X}]_{\eta_x}, u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$ with $\widetilde{G}_{D,\mathsf{q}_N}(x_\mathsf{q}, u_\mathsf{q}) \neq \varnothing$, it follows that $\widetilde{G}_{D,\mathsf{q}_{N+1}}(x_\mathsf{q}, u_\mathsf{q}) \subseteq \widetilde{G}_{D,\mathsf{q}_N}(x_\mathsf{q}, u_\mathsf{q})$. This implies that the relation $R = \left\{(x_\mathsf{q}, x'_\mathsf{q}) \in [\mathcal{X}]_{\eta_x} \times [\mathcal{X}]_{\eta_x} \mid x_\mathsf{q} = x'_\mathsf{q}\right\}$ is a 0-ASR from $\widetilde{G}_{D,\mathsf{q}_N}$ to $\widetilde{G}_{D,\mathsf{q}_{N+1}}$. Thus, from the assumption that R_D is the ε-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to S, R_D is the ε-ASR from $\widetilde{S}_{D,\mathsf{q}_{N+1}}$ to S. Therefore, it is inductively shown that R_D is the ε-ASR from $\widetilde{S}_{D,\mathsf{q}_N}$ to S for all $N \in \mathbb{N}_{>0}$.

Hence, any controller synthesized for the symbolic model $\widetilde{S}_{D,\mathsf{q}_N}$ can be refined to a controller for the original system S satisfying the same specification.

Remark 3 $\widetilde{S}_{D,\mathsf{q}_N}$ can have more (redundant) transitions than S_{D,q_N}, since the transitions of $\widetilde{S}_{D,\mathsf{q}_N}$ are updated only for some states, while in S_{D,q_N} these are updated for all states in $[\mathcal{X}]_{\eta_x}$. From Lemma 5, this implies that using S_{D,q_N} may result in a larger controlled invariant set than using $\widetilde{S}_{D,\mathsf{q}_N}$, which may be a drawback of using $\widetilde{S}_{D,\mathsf{q}_N}$. Nevertheless, as will be illustrated in the numerical example in the next section (Section 6), constructing $\widetilde{S}_{D,\mathsf{q}_N}$ should be more practical and useful than constructing S_{D,q_N}, since it achieves a significant reduction of the computational load. □

Remark 4 Let us mention that the use of lazy approaches has been previously used in the symbolic control literature (see the approaches proposed in [38,39,40] and a review of the lazy techniques in [41]). In these approaches, the refinement of the abstraction is done for the regions that are not able to achieve the safety specification (either by using finer discretizations or lower inputs). In this paper, the criteria of the refinement are different, since we are refining on the regions where we are able to collect new data. Moreover, the method of refinement is also different since we conserve the same discretizations, the same input, but we benefit from the supplementary knowledge on the un-modeled dynamics to reduce the redundant transitions. □

Algorithm 5 Derivation of $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$ for all $N \in \mathbb{N}_{\geq 1}$, $\ell \in \mathbb{N}_{\geq 0}$ (speeding up the computation of the predecessors).

Input: $\widetilde{S}_{D,\mathsf{q}_N}$, $\mathcal{Q}_{N,\ell}$, and $\mathcal{Q}_{N-1,\ell+1}$ (available if $N > 1$);
Output: $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$;
1: **if** $N = 1$ **then**
2: $\quad\mathcal{Q} \leftarrow \varnothing$, $\mathcal{Q}_{N-1,\ell+1} \leftarrow \varnothing$;
3: **else**
4: $\quad\mathcal{Q} \leftarrow \mathcal{Q}_{N-1,\ell+1}$;
5: **end if**
6: **for** each $x_\mathsf{q} \in \mathcal{Q}_{N,\ell} \backslash \mathcal{Q}_{N-1,\ell+1}$ **do**
7: $\quad$**for** each $u_\mathsf{q} \in [\mathcal{U}]_{\eta_u}$ **do**
8: $\quad\quad$**if** $\widetilde{G}_{D,\mathsf{q}_N}(x_\mathsf{q}, u_\mathsf{q}) \subseteq \mathcal{Q}_{N,\ell}$ **then**
9: $\quad\quad\quad\mathcal{Q} \leftarrow \mathcal{Q} \cup \{x_\mathsf{q}\}$;
10: $\quad\quad$**end if**
11: $\quad$**end for**
12: **end for**
13: $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell}) \leftarrow \mathcal{Q}$;

Let us now proceed by reducing the computational load of the predecessor operator $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}$ in order to speed up the safety controller synthesis. To this end, let $\mathcal{Q}_{N,\ell}$, $\ell = 0, 1, \ldots$ denote the sequence of sets $\mathcal{Q}_\ell$, $\ell = 0, 1, \ldots$ in Algorithm 1 by executing $\mathsf{SafeCon}(\widetilde{S}_{\mathsf{q}_N}, \mathcal{X})$. Then, it follows from $T_N \geq T_{N-1}$ for all $N \in \mathbb{N}_{>0}$ that $\mathcal{Q}_{N-1,\ell+1} \subseteq \mathcal{Q}_{N,\ell+1} = \mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$, for all $N \in \mathbb{N}_{>0}$ and $\ell \in \mathbb{N}_{>0}$ (see the proof of Lemma 5). Hence, when we aim at computing $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$, it is *known* that $\mathcal{Q}_{N-1,\ell+1}$ is the subset of $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$. This implies that all states in $\mathcal{Q}_{N-1,\ell+1}$ can be directly added to the predecessors for $\mathcal{Q}_{N,\ell}$ *without* checking the existence of a control input such that all successors are in $\mathcal{Q}_{N,\ell}$ according to (18).

Based on the above observation, we propose Algorithm 5 so as to speed up the computation of $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$. In the algorithm, $\mathcal{Q}$ represents the set of predecessors for $\mathcal{Q}_{N,\ell}$ which are mainly updated according to line 6–line 12. For $N = 1$, $\mathcal{Q}$ is initialized by the empty set (line 2). In other words, *all* states in $\mathcal{Q}_{N,\ell}$ (since $\mathcal{Q}_{N,\ell} \backslash \mathcal{Q}_{N-1,\ell+1} = \mathcal{Q}_{N,\ell}$) are evaluated to check the existence of a control input such that all the successors are in $\mathcal{Q}_{N,\ell}$ according to line 6 to line 12. For $N > 1$, on the other hand, $\mathcal{Q}$ is initialized by $\mathcal{Q}_{N-1,\ell+1}$ (line 4). This is due to the fact that it is *already known* that $\mathcal{Q}_{N-1,\ell+1}$ is a subset of the predecessors for $\mathcal{Q}_{N,\ell}$ (i.e., $\mathcal{Q}_{N-1,\ell+1} \subseteq \mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$). Hence, only the states in $\mathcal{Q}_{N,\ell} \backslash \mathcal{Q}_{N-1,\ell+1}$ (instead of $\mathcal{Q}_{N,\ell}$) are necessary to be evaluated to check the existence of a control input such that all the successors are in $\mathcal{Q}_{N,\ell}$ according to line 6–12.

In summary, during execution of $\mathsf{SafeCon}(\widetilde{S}_{\mathsf{q}_N}, \mathcal{X})$ (line 15 in Algorithm 2) for all $N \in \mathbb{N}_{\geq 1}$, $\mathrm{Pre}_{\widetilde{S}_{D,\mathsf{q}_N}}(\mathcal{Q}_{N,\ell})$,

$\ell = 0, 1, ...$ are computed by Algorithm 5. The computational complexity of computing predecessors according to Algorithm 5 for $N = 1$ is $\mathcal{O}(|\mathcal{Q}_{N,\ell}|\ |[\mathcal{U}]_{\eta_u}|)$. For $N > 1$, we have $\mathcal{O}(|\mathcal{Q}_{N,\ell} \backslash \mathcal{Q}_{N-1,\ell+1}|\ |[\mathcal{U}]_{\eta_u}|)$. Hence, the computation of the predecessors becomes faster as the cardinality of $\mathcal{Q}_{N,\ell} \backslash \mathcal{Q}_{N-1,\ell+1}$ becomes smaller, or in other words, $\mathcal{Q}_{N,\ell}$ is closer to $\mathcal{Q}_{N-1,\ell+1}$, i.e., $\mathcal{Q}_{N,\ell} \approx \mathcal{Q}_{N-1,\ell+1}$. Note that we have $\mathcal{Q}_{N,\ell} = \mathcal{Q}_{N-1,\ell+1}$ if the controlled invariant set converges $\mathcal{X}_{S,N-1} = \mathcal{X}_{S,N}$ (i.e., $\mathcal{Q}_{N,\ell} = \mathcal{Q}_{N-1,\ell}$, $\forall \ell \in \mathbb{N}_{\geq 0}$) and $\mathcal{Q}_{N,\ell} = \mathcal{Q}_{N,\ell+1}$ (i.e., $\mathcal{Q}_{N,\ell}$ converges to a fixed point). Hence, it is expected that Algorithm 5 becomes faster as both the controlled invariant set and $\mathcal{Q}_{N,\ell}$ get closer to their fixed points. The memory requirement is $\mathcal{O}(|[\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}|)$, since it needs to store the set $\mathcal{Q}_{N,\ell} \subseteq [\mathsf{Interior}_\varepsilon(\mathcal{X})]_{\eta_x}$.

6 Simulation results

In this section we illustrate the effectiveness of the proposed approach through a simulation of an adaptive cruise control (ACC) [42,43,44]. The simulation has been conducted on Windows 10, Intel(R) Core(TM) 2.40GHz, 8GB RAM. The state vector is given by $x = [x_1, x_2, x_3]^\mathsf{T} \in \mathbb{R}^3$, where x_1 is the velocity of the leading vehicle, x_2 is the velocity of the following vehicle, and x_3 is the distance between the lead vehicle and the following vehicle. Moreover, the input vector indicates the acceleration of the following car $u \in \mathbb{R}$. The dynamics is given by

$$x_{t+1} = x_t + \underbrace{\Delta \begin{bmatrix} 0 \\ u_t \\ x_{1,t} - x_{2,t} \end{bmatrix}}_{f(x_t,u_t)} + \underbrace{\Delta \begin{bmatrix} a_{l,t} \\ 0 \\ 0 \end{bmatrix}}_{v_t} + \underbrace{\Delta \begin{bmatrix} 0 \\ -(\nu_0 + \nu_1 x_{2,t} + \nu_2 x_{2,t}^2)/M \\ 0 \end{bmatrix}}_{d(x_t)},$$

where M is the weight of the following vehicles, Δ represents the sampling time, $a_{l,t}$ is the acceleration of the lead vehicle that is assumed to be the additive noise, and ν_0, ν_1, ν_2 are the constants for the aerodynamic drag force, whose function (i.e., $(\nu_0 + \nu_1 x_{2,t} + \nu_2 x_{2,t}^2)/M$) is assumed to be unknown apriori. It is assumed that $\Delta = 1, M = 1000, \nu_0 = 60, \nu_1 = 1.2, \nu_2 = 1.0$. Moreover, we assume that the velocity of the lead vehicle fulfills $15 \leq x_{1,t} \leq 25$ for all $t \in \mathbb{N}_{\geq 0}$, and its acceleration is bounded as $|a_{l,t}| \leq 0.2$ for all $t \in \mathbb{N}_{\geq 0}$. The safe set is given by $\mathcal{X} = \mathcal{Z} \backslash \mathcal{O}$, where $\mathcal{Z} = \{x \in \mathbb{R}^3 \mid 15 \leq x_1 \leq 25,\ 15 \leq x_2 \leq 25, 30 \leq x_3 \leq 80\}$, $\mathcal{O} = \{x \in \mathbb{R}^3 \mid 2x_2 \leq x_3\}$. The input constraint set is $\mathcal{U} = \{u \in \mathbb{R} \mid |u| \leq 1.0\}$. The initial state is given by

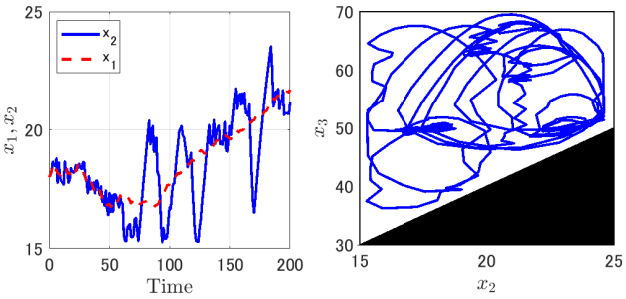

Fig. 1. The left figure illustrates the trajectories of x_1 and x_2 by Algorithm 2. The right figure illustrates the phase portrait in x_2, x_3 (the white region indicates the safe set $\mathcal{X}$).

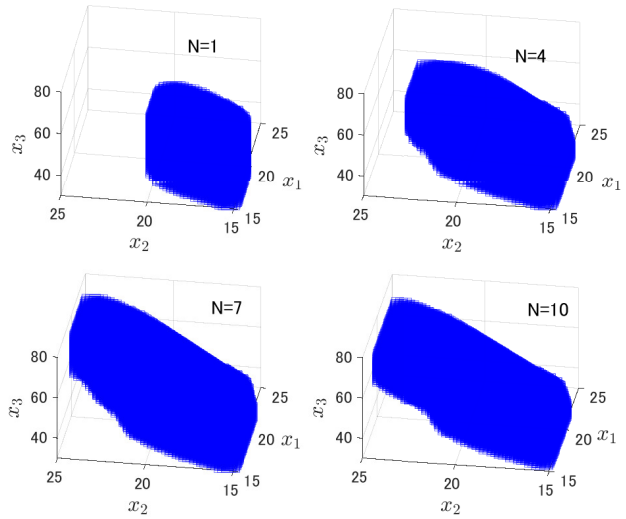

Fig. 2. The computed controlled invariant set $\mathcal{X}_{S,N}$ for $N = 1$ (upper left), $N = 4$ (upper right), $N = 7$ (lower left) and $N = 10$ (lower right).

$\bar{x} = [20, 20, 60]^\mathsf{T}$, and $\eta_x = \varepsilon = 0.2$, $\eta_u = 0.2$, $T_{\text{exp}} = 30$. Moreover, during the implementation of Algorithm 2, we incorporate Algorithm 4 and Algorithm 5 with $\rho = 0.01$ so as to reduce the computational load of abstractions and the safety controller synthesis. We used a squared-exponential $\mathsf{k}(x_t, x_{t'}) = \exp(-\alpha|x_t - x_{t'}|)$ with $\alpha = 1$. The computed upper bound of the RKHS norm was $\|d_2\|_\mathsf{k} \leq 2.0$ (for details on how to obtain this bound, see [34]).

For comparisons, we have also computed a symbolic model and a controlled invariant set by regarding $d_2(x_{2,t}) = \nu_0 + \nu_1 x_{2,t} + \nu_2 x_{2,t}^2/M$ as the *uniform disturbance* (i.e., the aerodynamic drag force will not be learned from data). We assume that the uniform disturbance satisfies $0.30 \leq d_2(x_{2,t}) \leq 0.71$, $\forall t \in \mathbb{N}$, since $\min_{x_2 \in [15,25]} d_2(x_2) = 0.30$ and $\max_{x_2 \in [15,25]} d_2(x_2) = 0.71$. These lower and the upper bounds of d_2 have

been utilized to construct the symbolic model and the controlled invariant set by following the abstraction procedure given in previous work, e.g., [45].

Fig. 1 shows the trajectories of x_1, x_2 by applying the proposed approach Algorithm 2 and the phase portrait of x_2, x_3. The figure illustrates that the trajectories are always inside $\mathcal{X}$ (white region), showing the achievement of the safe exploration. The algorithm terminates at $N = 10$. The computed controlled invariant sets for $N = 1, 4, 7, 10$ are illustrated in Fig. 2. The figure shows that the volume of the controlled invariant set is enlarged by collecting the training data according to Algorithm 2.

Fig. 3 shows the controlled invariant set finally obtained by applying the proposed approach (which is equivalent to the lower right of Fig. 2) and the uniform disturbance-based approach as described above. In addition, Fig. 4 shows $|[\mathcal{X}_{S,N}]_{\eta_x}|$ (i.e., the cardinality or the number of states contained in $[\mathcal{X}_{S,N}]_{\eta_x}$) by applying the proposed approach against the number of iterations (blue dotted line) and the uniform disturbance-based approach (green dotted line). Note that the size of the controlled invariant set under the uniform disturbance-based approach is constant for all the iterations, since the unknown function is not learned. The figure shows that the controlled invariant set obtained by the proposed approach becomes larger than the uniform disturbance-based approach after $N = 6$. This is because, by applying the proposed algorithm, the uncertainty of the unknown function becomes smaller as the iteration progresses, which results in reducing redundant transitions of the symbolic model (and thus enlarge the controlled invariant set); on the other hand, the uniform disturbance-based approach always considers the worst case effect of the disturbance, and thus the redundant transitions will not be removed. Moreover, for further comparisons, we also implemented Algorithm 2 *neither* by employing Algorithm 4 (i.e., S_{D,q_N} in (23) is constructed for each N) *nor* by employing Algorithm 5 for the safety controller synthesis, and the results are also plotted (red dotted lines). The right figure illustrates the total execution time to construct the symbolic model and the

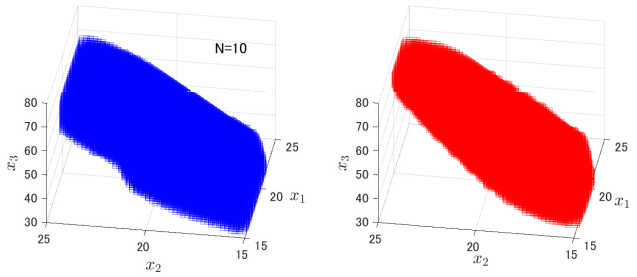

Fig. 3. The controlled invariant set finally obtained by applying the proposed approach (left) and the uniform disturbance-based approach (right).

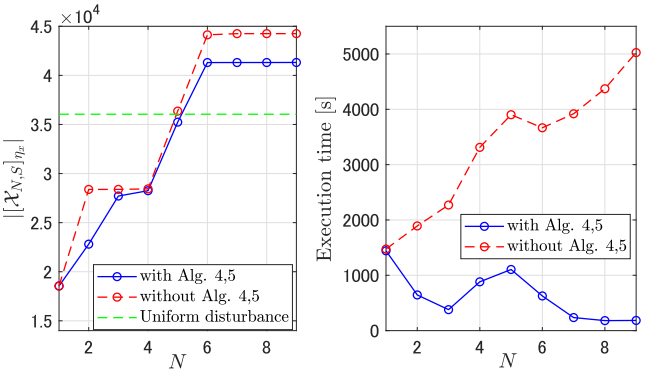

Fig. 4. The left figure indicates $|[\mathcal{X}_{S,N}]_{\eta_x}|$ with Algorithm 4 and 5 (blue solid), uniform disturbance-based approach (green dotted) and without Algorithm 4 and 5 (red dotted line). The right figure illustrates the total execution time to implement line 14 and line 15 in Algorithm 2.

safety controller (i.e., the execution time to implement line 14 and line 15 for each N in Algorithm 2). The figure implies that the controlled invariant set by constructing the symbolic model $\widetilde{S}_{D,\mathsf{q}_N}$ is smaller than by constructing S_{D,q_N}. As stated in Remark 3, this is due to the fact that in the former case the transitions are updated only for some states, while in the latter case these are updated for all states in $[\mathcal{X}]_{\eta_x}$. On the other hand, the total execution time (right figure in Fig. 4) by employing the former approach is shown to be significantly smaller than the latter approach, which illustrates the benefits of employing Algorithms 4 and 5.

Now, using the learned symbolic model $\widetilde{S}_{D,\mathsf{q}_N}$, we can synthesize a controller satisfying complex control specifications, such as *temporal logic formulas*. Following a correct-by-construction approach [42], we encode the requirements for the ACC by the linear temporal logic (LTL). First, consider two modes, called *set-speed mode* and *time-gap mode*. If the mode is in set-speed mode, the following vehicle must keep a given desired speed x_2^* with some accuracy, i.e., $|x_2 - x_2^*| \leq \epsilon_1$. If the mode is in time-gap mode, the following vehicle must achieve a desired time headway ω^* with some accuracy, i.e., $|x_3/x_2 - \omega^*| \leq \epsilon_2$. Let mode_1, mode_2 be atomic propositions, such that mode_1 (resp. mode_2) is satisfied if the mode is in set-speed mode (resp. the time-gap mode). It is assumed that mode_1 (resp. mode_2) is satisfied if the state is included in the set $\mathcal{X}_1 = \{x \in \mathcal{X} \mid x_3 \leq 60\}$ (resp. $\mathcal{X}_2 = \mathcal{X}_1 \backslash \mathcal{X}_1$). Let spec_1, spec_2 be the atomic propositions, such that spec_1 (resp. spec_2) is satisfied if $|x_2 - x_2^*| \leq \epsilon_1$ (resp. $|x_3/x_2 - \omega^*| \leq \epsilon_2$). Moreover, let safe be the atomic proposition, such that it is satisfied if the state is included in $\mathcal{X}$. Then, we encode the control specification by the LTL formula as follows:

$$\psi = \Box \mathsf{safe} \wedge \Box \wedge_{i=1}^{2} (\mathsf{mode}_i \implies \bigcirc \bigcirc \mathsf{spec}_i), \quad (25)$$

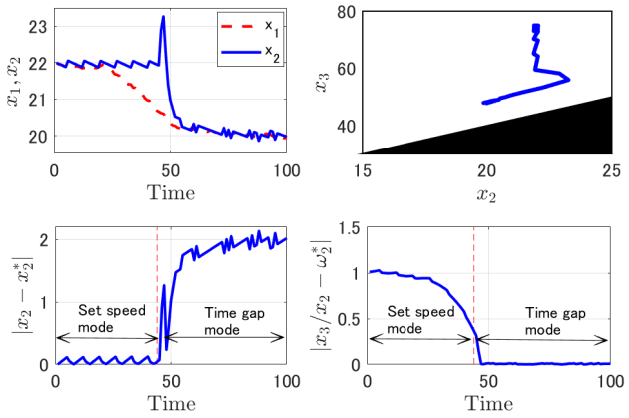

Fig. 5. The upper figures illustrate the trajectories of x_1 and x_2 (upper left) and the corresponding phase portrait in x_2, x_3 (upper right), by applying the synthesized controller satisfying ψ. The lower figures illustrate the absolute errors $|x_{2,t} - x_2^*|$ (lower left) and $|x_{3,t}/x_{2,t} - \omega^*|$ (lower right) with $x_2^* = 22$, $\omega^* = 2.4$.

where $\square$ and $\bigcirc$ are so-called the *"always"* and *"next"* temporal operators, respectively (see, e.g., [46]). In words, the state x must always stay in the safe set $\mathcal{X}$, and if the mode is in set-speed mode (resp. time-gap mode), the following vehicle must achieve the desired speed in two time steps (resp. the desired time headway in two time steps). Note that the controller for the safety specification $\square\mathsf{safe}$ has been already obtained after the implementation of Algorithm 2. The controller for the remaining part $\square \wedge_{i=1}^{2} (\mathsf{mode}_i \implies \bigcirc\bigcirc \mathsf{spec}_i)$ can be synthesized by a fixed point algorithm (see, e.g., [42]). The upper figures of Fig. 4 indicate the state trajectories by employing the synthesized controller with $x_2^* = 22, \omega^* = 2.4$ and $\epsilon_1 = 0.2, \epsilon_2 = 0.2$. Moreover, the lower figures indicate the sequences of the error $|x_2 - x_2^*|$ and $|x_3/x_2 - \omega^*|$. It can be verified that the formula ψ is satisfied by applying the synthesized controller, showing the effectiveness of the proposed approach.

7 Conclusions and future works

In this paper, we propose a learning-based approach towards symbolic abstractions for nonlinear control systems. The symbolic model is constructed by learning the un-modeled dynamics from training data, and the concept of an ε-approximate alternating simulation relation. Moreover, the safe exploration has been achieved by iteratively updating the controlled invariant and the safety controller, employing the safety game. In addition, we provide several techniques to alleviate the computational load to construct the symbolic models and the controlled invariant set. Finally, we illustrate the effectiveness of the proposed approach through a simulation example of an adaptive cruise control.

In our problem setup, it is of great importance to compute the upper bound of the RKHS norm $\|d_i\|_{\mathsf{k}_i} \leq B_i$ since it has been utilized to construct the symbolic models. Hence, as described in Section 6, obtaining a large enough, yet not too conservative bound for $\|d_i\|_{\mathsf{k}_i}$ should be further investigated in the future. In addition, since there exist no outliers in our problem setup, investigating how these can affect (if they exist) the estimation accuracy of the unknown function as well as how to detect them should be further pursued in future work.

References

[1] A. S. Seshia. New frontiers in formal methods: Learning, cyber-physical systems, education, and beyond. *CSI Journal of Computing*, 2(4), 2015.

[2] Giordano Pola and Maria Domenica Di Benedetto. Control of cyber-physical-systems with logic specifications: a formal methods approach. *Annual Reviews in Control*, 47:178–192, 2019.

[3] P. Tabuada. *Verification and Control of Hybrid Systems – A Symbolic Approach*. Springer, 2009.

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

[5] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[6] 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, 2010.

[7] G. Pola and P. Tabuada. Symbolic models for nonlinear control systems: Alternating approximate bisimulations. *SIAM Journal on Control and Optimization*, 48(2):719–733, 2009.

[8] M. Zamani, G. Pola, M. Mazo Jr., and P. Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2012.

[9] P. J. Meyer and D. V. Dimarogonas. Compositional abstraction refinement for control synthesis. *Nonlinear Analysis: Hybrid Systems*, 27:437–451, 2018.

[10] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. V. Dimarogonas. A symbolic approach to the self-triggered design for networked control systems. *IEEE Control Systems Letters*, 3(4):1050–1055, 2019.

[11] M. Rungger and P. Tabuada. A notion of robustness for cyber-physical systems. *IEEE Transactions on Automatic Control*, 61(8):2108–2123, 2016.

[12] M. Mizoguchi and T. Ushio. Deadlock-free output feedback controller design based on approximately abstracted observers. *Nonlinear Analysis: Hybrid Systems*, 30:59–71, 2018.

[13] M. Khaled, K. Zhang, and M. Zamani. Output-feedback symbolic control. *https://arxiv.org/abs/2011.14848*, 2020.

[14] A. S. Seshia, D. Sadigh, and S. S.Sastry. Towards verified artificial intelligence. *https://arxiv.org/pdf/1606.08514.pdf*, 2016.

[15] C. F. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[16] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

[17] F. Berkenkamp, R. Moriconi, A.P.Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *Proceedings of the IEEE 55th Conference on Decision and Control (IEEE CDC)*, pages 4661–4666, 2016.

[18] F. Berkenkamp, M. Turchetta, A.P.Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, page 908–919, 2017.

[19] P. J. Meyer, A. Girard, and E. Witrant. Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*, 63(6):1835–1841, 2018.

[20] P. Tabuada. An approximate simulation approach to symbolic control. *IEEE Transactions on Automatic Control*, 53(6):1406–1418, 2008.

[21] J. Jackson, L. Laurenti, E. W. Frew, and M. Lahijanian. Safety verification of unknown dynamical systems via gaussian process regression. In *Proceedings of the IEEE 59th Conference on Decision and Control (IEEE CDC)*, pages 860–866, 2020.

[22] G. Chen, P. Wei, and M. Liu. Temporal logic inference for fault detection of switched systems with gaussian process dynamics. *IEEE Transactions on Automation Science and Engineering*, 2021.

[23] J. Umlauft and S. Hirche. Feedback linearization based on gaussian processes with event-triggered online learning. *IEEE Transactions on Automatic Control*, 65(10):4154–4169, 2019.

[24] T. Beckers, D. Kulic, and S. Hirche. Stable gaussian process based tracking control of euler-lagrange systems. *Automatica*, 103:390–397, 2019.

[25] J. Umlauft, L. Pohler, and S. Hirche. An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process. *IEEE Control Systems Letters*, 2(3):483–488, 2018.

[26] K. Hashimoto, Y. Yoshimura, and T. Ushio. Learning self-triggered controllers with gaussian processes. *IEEE Transactions on Cybernetics*, 2021.

[27] L. Wang, E. A. Theodorou, and M. Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *Proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA 2018)*, pages 2460–2465, 2018.

[28] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin. Reachability-based safe learning with gaussian processes. In *Proceedings of 53rd IEEE Conference on Decision and Control*, pages 1424–1431, 2014.

[29] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2019.

[30] V. Dhiman, M. J. Khojastech, M. Franceschetti, and N. Atanasov. Control barriers in bayesian learning of system dynamics. *https://arxiv.org/pdf/2012.14964.pdf*, 2020.

[31] P. Jagtap, G. J. Pappas, and M. Zamani. Control barrier functions for unknown nonlinear systems using gaussian processes. In *Proceedings of the 59th IEEE International Conference on Decision and Control (CDC)*, pages 3699–3704, 2020.

[32] A. Devonport, H. Yin, and M. Arcak. Bayesian safe learning and control with sum-of-squares analysis and polynomial kernels. In *Proceedings of the 59th IEEE International Conference on Decision and Control (CDC)*, pages 3159–3165, 2020.

[33] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration. In *Proceedings of the IEEE 55th Conference on Decision and Control (IEEE CDC)*, pages 6059–6066, 2018.

[34] K. Hashimoto, A. Saoud, M. Kishida, T. Ushio, and D. V. Dimarogonas. Learning-based symbolic abstractions for nonlinear control systems. *in arxiv, available on https://arxiv.org/pdf/2004.01879.pdf*, 2022.

[35] P. Scharnhorst, E. T. Maddalena, Y. Jiang, and C. N. Jones. Robust uncertainty bounds in reproducing kernel hilbert spaces: A convex optimization approach. *https://arxiv.org/pdf/2104.09582.pdf*, 2021.

[36] C. A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *The Journal of Machine Learning Research*, 7:2651–2667, 2006.

[37] F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[38] A. Girard, G. Gossler, and S. Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Transactions on Automatic Control*, 61(6):1537–1549, 2016.

[39] O. Hussien and P. Tabuada. Lazy controller synthesis using three-valued abstractions for safety and reachability specifications. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3567–3572, 2018.

[40] Z. Kader, A. Saoud, and A. Girard. Safety controller design for incrementally stable switched systems using event-based symbolic models. In *Proceedings of 2019 European Control Conference (ECC 2019)*, pages 1269–1274, 2019.

[41] A. Saoud. *Compositional and Efficient Controller Synthesis for Cyber-Physical Systems*. Ph.D. Thesis, 2019.

[42] 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, 2016.

[43] A. D. Ames, X. Xu, J. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[44] 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*, 2019.

[45] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2017.

[46] C. Baier and J.-P Katoen. *Principles of model checking*. The MIT Press, 2008.