PAC Model Checking of Black-Box Continuous-Time Dynamical Systems

Bai Xue, Member, IEEE, Miaomiao Zhang, Arvind Easwaran, Senior Member, IEEE, and Qin Li

Abstract—In this article, we present a novel model checking approach to finite-time safety verification of black-box continuous-time dynamical systems within the framework of probably approximately correct (PAC) learning. The black-box dynamical systems are the ones, for which no model is given but whose states changing continuously through time within a finite-time interval can be observed at some discrete-time instants for a given input. The new model checking approach is termed as the PAC model checking due to the incorporation of learned models with correctness guarantees expressed using the terms error probability and confidence. Based on the error probability and confidence level, our approach provides statistically formal guarantees that the time-evolving trajectories of the black-box dynamical system over finite-time horizons fall within the range of the learned model plus a bounded interval, contributing to insights on the reachability of the black-box system and thus on the satisfiability of its safety requirements. The learned model together with the bounded interval is obtained by scenario optimization, which boils down to a linear programming problem. Three examples demonstrate the performance of our approach.

Index Terms—Black-box dynamical systems, linear programming, probably approximately correct (PAC) model checking.

I. INTRODUCTION

The complexity of today’s technological applications induces a quest for automation, leading to many black-box intelligent cyber–physical systems and thus being difficult to reason about [25]. Many of these systems operate in a safety-critical context and hence safety-critical systems themselves [32]. Therefore, reasonable performance guarantees should be obtained before the systems are deployed.

Black-box checking, introduced by Peled et al. [31], is often used for verifying nonstochastic black-box systems, based on experiments that interface with them. It performs checks on the system itself. The black-box checking is a combination of model checking and testing: model checking [12] checks properties of a model of the system, but not the system itself. In the contrary, testing is usually applied to the actual system and checks whether the system conforms with the model, further serving to improve the model. They are two complementary approaches for enhancing the reliability of black-box systems. In the black-box checking, whenever a model is created, model checking may reveal a fault in the system or show that the model was not good enough and needs to be learned further if the fault is spurious. If model checking does not reveal a fault, equivalence between the model and the black-box system is checked via testing. In case, nonequivalence is detected, then the model needs to be further learned. The checking–testing–learning repeated process is costly generally. Recently, a method combining optimization-based falsification and black-box checking was proposed to falsify specifications for black-box cyber–physical systems in [40].

Another technique to verification of black-box systems is statistical model checking (SMC) [35], [45]. SMC is pioneered by Younes and Simmons in the discrete case in [47], which is based on the sequential probability ratio test [41]. It is a compromise between verification and testing, which is based on sampling executions of the system and then deciding whether the samples provide statistical evidence for the satisfaction or violation of the specification based on hypothesis testing [34]. SMC is now widely accepted in various research areas such as software engineering, in particular, for industrial applications [13], or even for solving problems originating from systems biology [11]. There are several reasons for this success. First, SMC is very simple to understand, implement, and use. Second, it does not require extra modeling or specification effort, but simply an executable system that can be simulated and checked against state-based properties. Third, it avoids the state space explosion in verification and thus can be applied to analyze systems with large state spaces. Consequently, there are variety of SMC tools, such as PLASMA-Lab [3], Ymer [46], VeStA [36], MRMC [24], MC2 [20], UPPAAL-SMC [14], and so on. In order to further improve the efficiency of SMC, Bayesian SMC was proposed in [23] and [48], which is an SMC based on Bayesian statistics. The aforementioned SMC approaches for
black-box systems are free of mathematical models and perform checks on the system itself by sampling executions of the system. However, the usefulness of mathematical models is well documented. The mathematical models not only help us to understand the system but also are instrumental to yield insight into the complex processes involved in the system by extracting the essential meaning of some hypotheses. Also, they allow us to study the effects of changes in their components and/or environmental conditions on the system’s trajectories, i.e., they allow the control and optimization of the system. Thus, the introduction of mathematical models with an appropriate degree of complexity into SMC would contribute a lot to the analysis of the black-box system, not only in the verification of its specifications but also in understanding the complex mechanisms underlying and thus further optimizing the system. Consequently, model learning-based SMC approaches are also proposed. For example, Aichernig and Tapler [1] and Mao et al. [26], [28] considered black-box systems modeled by Markov decision processes and inferred probabilistic models with the purpose of model checking. The work in [29] combined stochastic learning and abstraction with respect to some property for analyzing black-box systems modeled by Markov decision processes. The work in [4] presented an approach for black-box systems modeled by Markov decision processes to unbounded reachability analysis via SMC. The technique is based on delayed Q-learning, a form of reinforcement learning. Generally, the exact learning algorithms require checking equivalence between the model and the system, which is difficult and undecidable. Regression models were used in [17] for finding the regions in the parameter space that lead to satisfaction or violation of given specification with probabilistic coverage guarantees based on conformal regression. Recently, learning procedure within the probably approximately correct (PAC) learning framework is proposed, e.g., [2], [10], [19], and [30].

In this article, we propose a novel SMC approach for finite-time safety verification of black-box continuous-time dynamical systems within the framework of PAC learning [18]. The black-box continuous-time dynamical systems are the ones, for which no model is given but whose states changing continuously through time over finite-time horizons can be observed at some discrete-time instants for a given input. The proposed new model checking, also termed as the PAC model checking, is built upon learned models within the framework of PAC learning. In the PAC model checking, correctness guarantees of the learned models are expressed using the terms error probability and confidence level. We show that the time-evolving trajectories of the black-box system over a specified finite-time horizon fall within the range of the learned model plus a bounded interval with statistical guarantees, which is further used to characterize the satisfiability of safety requirements. Given an error probability and a confidence level, which are two fundamental parameters in PAC learning, the model together with the bounded interval is computed via scenario optimization, which is widely used for computing solutions to robust optimization problems based on finite randomization of infinite constraints [5]. The scenario optimization, which finally boils down to a linear program in our approach, is constructed from a family of independent and identically distributed datum collected by executing the system. Three examples demonstrate the performance of our approach. Our contributions are summarized as follows.

1) We propose a novel PAC model checking approach for finite-time safety verification of black-box continuous-time dynamical systems. In this approach, the trajectories of the black-box system over finite-time horizons are shown to fall within the range of a model plus a bounded interval with error probabilities and confidence levels. This reachability analysis is instrumental in characterizing the satisfiability of safety requirements of the black-box system.

2) A linear programming-based approach is proposed to synthesize the model and the bounded interval. The size of the linear programming problem could be independent of the one of the black-box system, thus rendering our approach suitable for large-scale systems.

Related Work: As mentioned above, there are many works on verifying black-box systems. In this section, we just discuss the closely related works to the present one.

The works [2] and [19] considered (unbounded) reachability for Markov decision processes (and stochastic games in [2]) and inferred the transition probabilities with PAC guarantees. The work [30] proposed an algorithm for constructing PAC confidence sets for deep neural networks. The work in [43] computed safe inputs for a black-box system such that the system’s final outputs fall within a safe range with PAC guarantees. In contrast, our approach focuses on the analysis of continuous-time systems and infers that the time-evolving trajectories of the black-box system over finite-time horizons fall within the range of a model plus a bounded interval with PAC guarantees. The closest work in spirit to the present one is [10], which considered the verification of sequential programs by learning models of the set of feasible paths of programs within the framework of PAC learning. The model learning algorithm in [10] is based on counterexample guided abstraction refinement. However, our approach considers continuous-time systems and infers an approximation to the trajectories of the system over the specified finite-time horizon within the framework of PAC learning, in which linear programs are used for learning models.

In the framework of simulation-driven reachability analysis [15], a PAC-based method was proposed for learning discrepancy functions in [16] for safety verification of hybrid systems with black-box modules. The problem of learning discrepancy functions is reduced to a problem of learning linear separators. Although a PAC discrepancy function is computed in [16], a characterization on how well the trajectories satisfy the learned discrepancy function is not given, and thus a formal quantitative assessment on the satisfiability of safety properties is not presented if a valid discrepancy function is not obtained. Generally, valid discrepancy functions rather than PAC ones for black-box systems are challenging to obtain. In contrast, a formal characterization of the satisfiability of safety properties is given based on the computation of PAC models in our PAC model checking method.

When the continuous-time systems of interest are modeled by ordinary differential equations or delay differential equations, and the equations are explicitly given, there
are many well-developed model-based reachability analysis techniques over finite-time horizons, e.g., Taylor-model method [9], simulation-driven reachability method [15], and set-boundary reachability method [44], for safety verification of these systems. However, our method focuses on black-box continuous-time dynamical systems, whose mathematical abstractions are not acquired and which are only represented by a family of datum. Such systems cannot be handled by existing model-based reachability analysis techniques.

The remainder of this article is structured as follows. In Section II, we formalize the concept of black-box continuous-time dynamical systems and the problem of interest in this article. Section III elucidates our PAC model checking approach. After demonstrating the performance of our approach on three examples in Section IV, we conclude this article in Section V.

II. PRELIMINARIES

In this section, we present the concept of black-box continuous-time dynamical systems and the related problems, as well as a brief introduction on scenario optimization. The notations are used throughout this article: $\mathbb{R}_{\geq 0}$ denotes the set of nonnegative real values, $\mathbb{R}_{> 0}$ denotes the set of positive real values. Vectors are denoted by boldface letters. Besides, the ground-truth trajectories in all examples are obtained based on the combination of Runge–Kutta simulation methods and linear interpolation methods.

A. Problem Formulation

In this article, we consider a black-box continuous-time dynamical system, whose dynamics are governed by a formula of the following form:

$$y(t) = b(x_0, t) \quad (1)$$

where $x_0 = (x_{0,1}, \ldots, x_{0,p})^\top \in \mathcal{X}_0$ is the input of the system, the set $\mathcal{X}_0 \subseteq \mathbb{R}^p$ is compact, $t \in [0, T]$ with $T \in \mathbb{R}_{> 0}$ is the time variable, $y(t)$ is the state of the system at time $t$, and $b(\cdot, \cdot) : \mathcal{X}_0 \times [0, T] \rightarrow \mathbb{R}$ is the system mapping which is unknown. Besides, we have the following assumptions.

Assumption 1:
1) The system (1) runs well, including the onboard sensors, and thus it can provide us any family of finite datum we need. Also, the provided datum is free of noise.
2) Suppose that the time horizon $[0, T]$ is endowed with a $\sigma$–algebra $\mathcal{D}_t$ and a probability $P_t$ over $\mathcal{D}_t$ is assigned. Also, we assume that the set $\mathcal{X}_0$ of inputs is endowed with a $\sigma$–algebra $\mathcal{D}_{x_0}$ and that a probability $P_{x_0}$ over $\mathcal{D}_{x_0}$ is assigned. Throughout this article, we use the uniform distribution $P_t$ on $[0, T]$ and $P_{x_0}$ on $\mathcal{X}_0$ to illustrate our method, although our method is not confined to this particular distribution.

The system (1) is illustrated in Fig. 1. Given an input $x_0 \in \mathcal{X}_0$, the trajectory of the system (1) with the input $x_0$ is denoted by $y_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R}$.

Systems of the form (1) are all around us, especially nowadays. For example, many AI systems, such as robotics and self-driving cars, are leaving academic laboratories and entering real-world applications. Unfortunately, many of these systems cannot explain their results even to their makers, let alone to end users [7]. They operate like black boxes, which can be viewed in terms of a family of observed datum, without any knowledge of their internal workings.

In this article, we propose a PAC model checking approach for finite-time safety verification of the system (1). The safety verification problem is widely studied in computer science, e.g., [22]. In our approach, the key is to obtain a model with an appropriate degree of complexity, which is learned based on a family of collected datum within the framework of PAC learning and can characterize the system (1) with correctness guarantees expressed with error probabilities and confidence levels. For computing such models, we should address the problems summarized as follows.

Problem 1:
1) What datum should we use?
2) How can we learn a mathematical model efficiently based on the collected datum?
3) What is the discrepancy between the trajectories of the learned mathematical model and the system (1)?

After computing the model, we will address the safety verification problem as follows.

Problem 2: Given a set $\mathcal{Un}_s \subseteq \mathbb{R}$ of unsafe states, when the trajectories of the computed model are shown to avoid the set $\mathcal{Un}_s$, how can we formally characterize the satisfiability of the safety property of avoiding the unsafe set $\mathcal{Un}_s$ for the black-box system (1) over the time horizon $[0, T]$?

We in the sequel solve Problems 1 and 2 based on scenario optimization.

Remark 1: Our method can be straightforwardly extended to vector valued mappings of the form $b(\cdot, \cdot) : \mathcal{X}_0 \times [0, T] \rightarrow \mathbb{R}^q$ with $q > 1$, but the scalar valued mappings $b(\cdot, \cdot) : \mathcal{X}_0 \times [0, T] \rightarrow \mathbb{R}$ are considered for ease of exposition.

B. Scenario Optimization

This section gives a brief introduction on scenario optimization. It provides statistical solutions to robust optimization problems based on solving finite randomization of infinite convex constraints.

A robust optimization problem of interest is as follows:

$$\min_{\gamma \in \Gamma} \mathbb{E}[\mathbb{R}^m] e^\top \gamma$$

s.t. $f_\delta(\gamma) \leq 0 \forall \delta \in \Delta$ \quad (2)

where $f_\delta(\gamma)$ are continuous and convex functions over the $m$-dimensional optimization variable $\gamma$ for every $\delta \in \Delta$. Also, the sets $\Gamma$ and $\Delta$ are convex and closed.
Generally, it is challenging to solve (2). The work in [5] proposed a scenario optimization approach for solving (2) with statistically formal guarantees.

Definition 1: Suppose that \( \Delta \) is endowed with a \( \sigma \)-algebra \( \mathcal{D} \) and that a probability \( P \) over \( \mathcal{D} \) is assigned. The scenario optimization of (2) is to obtain an approximate solution to (2) via solving the convex program (3), which is constructed by extracting \( K \) independent and identically distributed samples \( \{\delta_i\}_{i=1}^{K} \) from \( \Delta \) according to the probability distribution \( P \)

\[
\begin{align*}
\min_{\gamma \in \Gamma \subseteq \mathbb{R}^m} & \quad c^T \gamma \\
\text{s.t.} & \quad \delta^T \gamma \leq 0.
\end{align*}
\] (3)

Equation (3) relaxes (2) in that it only considers a finite subset of the infinitely many constraints of (2). A mathematically rigorous relation, which holds irrespective of the underlying probability \( P \), between the solutions of the two systems can be drawn [6].

Theorem 1: If (3) is feasible and attains a unique optimal solution \( \gamma^*_k \), and

\[
\epsilon \geq \frac{2}{K} \left( \ln \frac{1}{\beta} + m \right)
\] (4)

where \( \epsilon \in (0, 1) \) and \( \beta \in (0, 1) \) are, respectively, a user-chosen error level and confidence level, then with at least \( 1 - \beta \) confidence, \( \gamma^*_k \) satisfies all constraints in \( \Delta \) but at most a fraction of probability measure \( \epsilon \), i.e., \( P(\delta \in \Delta | \delta^T \gamma^*_k \leq 0) \leq \epsilon \), where the confidence \( \beta \) is the \( K \)-fold probability \( P^K \) in \( \Delta^K = \Delta \times \cdots \times \Delta \), which is the set to which the extracted sample \( \{\delta_1, \ldots, \delta_K\} \) belongs.

The above conclusion still holds if the uniqueness of optimal solutions to (3) is removed [5], since a unique optimal solution can always be obtained according to the Tie-break rule if multiple optimal solutions occur. Moreover, since \( \beta \) appears under the sign of logarithm in (4), it can be made small, like \( 10^{-10} \) or \( 10^{-20} \), without increasing \( K \) significantly. Recently, scenario optimization was used to compute approximately safe inputs for a black-box system such that the system’s final outputs fall within a safe range in [43], and perform safety verification of hybrid systems in [42].

III. PAC MODEL CHECKING

In this article, we present our PAC model checking approach for safety verification of the black-box system (1) by solving Problems 1 and 2.

A. Datum Extraction

In this section, we introduce what datum to use in learning a model of the system (1) in our approach and how to obtain them, i.e., solve Problem 1 1).

We first extract a family of independent and identically distributed time instances \( \{(t_j^M)_{j=1}^{M}\} \) from the time interval \([0, T]\) according to the probability distribution \( P_j \). Moreover, a family of independent and identically distributed inputs \( \{x_{0,i,j}\}_{j=1}^{M} \) is also extracted from the set \( \mathcal{X}_0 \) according to the probability distribution \( P_{X_0} \). The process of obtaining \( \{(t_j^M)_{j=1}^{M}\} \) and \( \{x_{0,i,j}\}_{j=1}^{M} \) does not need to run/simulate the system (1). The numbers \( M \) and \( N \) rely on how accurate one wants the learned model to achieve. The relationship is elucidated in Section III-B.

Next, we need to run the system (1) to obtain its internal datum. For each extracted input \( x_{0,i} \), \( i = 1, \ldots, N \), feed it to the system (1) and then run it until the time \( T \). In this process, the onboard sensors will help observe and record the states of the system (1) at the time instance \( t_j, j = 1, \ldots, M \). This is realistic for some systems nowadays since smart sensors are taking over almost every sphere of human life. For example, RADAR, LIDAR, GPS, and computer vision are widely used to work coherently for identifying the position, velocity, and other states of the vehicle. We denote the family of observed states by \( \{(y_{i,j})_{i=1,\ldots,N,j=1,\ldots,M}\} \), where \( y_{i,j} \) denotes the state of the system (1) at time \( t_j \) with the input \( x_{0,i} \), \( i = 1, \ldots, N \), \( j = 1, \ldots, M \).

So far, we obtain a family of datum \( \{(x_{0,i}, t_j, y_{i,j})\}_{i=1,\ldots,N,j=1,\ldots,M} \). Each data are a triple \( (x_{0,i}, t, y(t)) \), where \( x_{0,i} \) is the input of the system (1), \( t \in [0, T] \) is the time instance, and \( y(t) \) is the state of the system (1) with the input \( x_{0,i} \) at time \( t \). The process of running the system (1) can be regarded as a testing process. However, our method goes further than testing techniques. We meanwhile collect a family of datum and then use these datum to compute models for characterizing the system (1) formally.

In our experiment, we assume that the input \( x_{0,i} \) is noise free and the onboard sensors work perfectly such that the observed datum are free of noise as well, i.e., \( y_{i,j} \) is the exact state of the system (1) with the input \( x_{0,i} \) at time \( t = t_j, i = 1, \ldots, N, j = 1, \ldots, M \). This assumption may be too ideal in practice since input and sensor noise often exist. We would relax it in our future work.

B. Safety Verification

In this section, we elucidate our approach for solving Problems 1 2), 1 3), and 2 based on the family of datum obtained from the process in Section III-A. We first consider the system (1) with one trajectory, and then multiple trajectories and finally all trajectories from the input set \( X_0 \).

1) One Trajectory Verification: In this section, we solve Problems 1 2), 1 3), and 2 for the system (1) with a single input. Concretely, given a discrete-time trajectory of the system (1) with the input \( x_{0,i} \), which is represented by a family of datum \( \{(x_{0,i}, t_j, y_{i,j})\}_{j=1}^{M} \) with \( (t^M_j)_{j=1}^{M} \) and \( (y_{i,j})_{j=1}^{M} \) obtained in Section III-A, we would compute a model \( z(t) = w(x_{0,i}, t) \) with \( w(x_{0,i}, \cdot) : [0, T] \rightarrow \mathbb{R} \) to characterize \( y_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R} \).

PAC Models: In computing a model, we consider a linearly parameterized model template \( w(c_1, \ldots, c_k, x_{0,i}, t) \), \( k \geq 1 \) such that \( w(c_1, \ldots, c_k, x_{0,i}, t) \) is for \( t \in [0, T] \) a linear function in \( c_1, \ldots, c_k \), which are unknown parameters. This model can be a polynomial function over \( t \) or a more general nonlinear function over \( t \). For instance, consider a 2-D system with input state variable \( x = (x_1, x_2)^T \), \( w(c_1, c_2, x, t) = c_1 x_1 t + c_2 x_2 t^2 \) is a linear function in \( c_1 \) and \( c_2 \), and \( w(c_1, c_2, x, t) = c_1 e^{t x_1 x_2} + c_2 \ln (x_2 t^2) \) is also a linear function over \( c_1 \) and \( c_2 \). Such models can be the ones parameterized with orthonormal basis functions, which are able to represent a set of physical systems [21]. For ease of exposition, we use \( c \) to
denote \((c_l)_{l=1,\ldots,k}\) in the reminder of this article. Generally, a model template of appropriate degree of complexity should be chosen in order to avoid the overfitting issue and facilitate the reachability analysis. In practice, engineering insight and physical knowledge would facilitate the selection of model templates.

Then, we construct the following linear program over \(c\) for computing a mathematical model based on the family of given datum \(((x_{0,i,j}, t_j, y_{i,j}))_{j=1}^M\):

\[
\begin{align*}
\min_{c, \xi} & \quad \xi \\
\text{s.t.} & \quad \text{for each } j = 1, \ldots, M: \\
& \quad w(c, x_{0,i,j}, t_j) - b(x_{0,i,j}, t_j) \leq \xi \\
& \quad b(x_{0,i,j}, t_j) - w(c, x_{0,i,j}, t_j) \leq \xi \\
& \quad -U_c \leq c_l \leq U_c, \quad l = 1, \ldots, k \\
& \quad 0 \leq \xi \leq U_\xi
\end{align*}
\]

which is equivalent to

\[
\begin{align*}
\min_{c, \xi} & \quad \xi \\
\text{s.t.} & \quad \text{for each } j = 1, \ldots, M: \\
& \quad w(c, x_{0,i,j}, t_j) - y_{i,j} \leq \xi \\
& \quad y_{i,j} - w(c, x_{0,i,j}, t_j) \leq \xi \\
& \quad -U_c \leq c_l \leq U_c, \quad l = 1, \ldots, k \\
& \quad 0 \leq \xi \leq U_\xi
\end{align*}
\]

where \(U_c \in \mathbb{R}_{>0}\) is a prespecified upper bound for \(c_l, \quad l = 1, \ldots, k\), and \(U_\xi \in \mathbb{R}_{>0}\) is a prespecified upper bound for \(\xi\).

Denote the optimal solution to (6) by \((c^*, \xi^*)\). Thus, we obtain a model \(z(t) = w(c^*, x_{0,i})\), whose discrepancy with the system (1) is characterized by two approximation parameters: 1) error probability \(\epsilon \in (0, 1)\) and 2) confidence level \(\beta \in (0, 1)\). This is formally stated in Theorem 2.

**Theorem 2:** Let \((c^*, \xi^*)\) be an optimal solution to (6), \(\epsilon \in (0, 1)\), \(\beta \in (0, 1)\), and

\[
\epsilon \geq 2 \left( \ln \frac{1}{\beta} + k + 1 \right). \tag{7}
\]

Then, we have that with at least \(1 - \beta\) confidence

\[
P_c\left( \{ t \in [0, T] \mid |w(c^*, x_{0,i}) - b(x_{0,i}, t)| \leq \xi^* \} \right) \geq 1 - \epsilon. \tag{8}
\]

**Proof:** The conclusion is easily obtained by Theorem 1. □

Actually, the computed mathematical model \(z(t) = w(c^*, x_{0,i})\) is a PAC model [37], [38] with accuracy level \(\epsilon\) and confidence level \(\beta\). The accuracy parameter \(\epsilon\) in Theorem 2 determines how far the learned model can be from the real one. This corresponds to the “approximately correct.” A confidence parameter \(\beta\) indicates how likely the learned model is to meet that accuracy requirement. This corresponds to the “probably” part. Under the data access model that we are investigating, these approximations are inevitable. Since the training set \(((x_{0,i,j}, t_j, y_{i,j}))_{j=1}^M\) is randomly generated, there may always be a small chance that it will happen to be non-informative (for example, there is always some chance that the training set will contain only one domain point, sampled over and over again). Furthermore, even when we are lucky enough to get a training sample that does faithfully represent [0, T], because it is just a finite sample, there may always be some finite details of [0, T] that it fails to reflect. The accuracy parameter \(\epsilon\) allows forgiving the learned model for making minor errors.

**One Trajectory Verification:** Based on Theorem 2, we in this section solve Problem 2 for the system (1) with one trajectory \(y_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R}\) using the trajectory of the mathematical model \(z(t) = w(c^*, x_{0,i}, t)\) within the framework of PAC learning.

We first characterize the reachability of the trajectory \(y_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R}\) using the mathematical model \(z(t) = w(c^*, x_{0,i}, t)\) plus the computed \(\xi^*\). We denote the trajectory of the mathematical model \(z(t) = w(c^*, x_{0,i}, t)\) by \(z_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R}\). From Theorem 2, we have that with confidence of at least \(1 - \beta\)

\[
y_{x_{0,i}}(t) \in [z_{x_{0,i}}(t) - \xi^*, z_{x_{0,i}}(t) + \xi^*] \tag{9}
\]

for all \(t \in [0, T]\) but at most a fraction of probability measure \(\epsilon\), i.e., with confidence of at least \(1 - \beta\), the amount of time for the trajectory \(y_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R}\) staying within the \(\xi^*\)-neighborhood of the trajectory \(z_{x_{0,i}}(\cdot) : [0, T] \rightarrow \mathbb{R}\) exceeds \(T(1 - \epsilon)\). A graph explanation is further presented in Fig. 2 to enhance the understanding of (9). In Fig. 2, \(y_{x_{0,i}}(t) \notin [z_{x_{0,i}}(t) - \xi^*, z_{x_{0,i}}(t) + \xi^*]\) for \(t \in [t_1, t_2] \cup [t_3, t_4] \cup [t_5, t_6]\). According to Theorem 2, \(t_6 - t_5 + t_4 - t_3 + t_2 - t_1 \leq \epsilon T\) with confidence of at least \(1 - \beta\).

Then, we solve Problem 2 based on the formal reachability characterization given above. That is, if \([z_{x_{0,i}}(t) - \xi^*, z_{x_{0,i}}(t) + \xi^*]\) does not intersect the unsafe set \(\mathcal{U}_{\mathcal{N}}\) for \(t \in [0, T]\), i.e., \([z_{x_{0,i}}(t) - \xi^*, z_{x_{0,i}}(t) + \xi^*] \cap \mathcal{U}_{\mathcal{N}} = \emptyset\) for \(t \in [0, T]\), we have that the amount of time the system (1) with the input \(x_{0,i}\) spends inside the unsafe set \(\mathcal{U}_{\mathcal{N}}\) does not exceed \(\epsilon T\), with confidence of at least \(1 - \beta\).

If \(\beta\) in Theorem 2 is extremely small (smaller than \(10^{-20}\)), then we have a priori practical certainty that the total amount of unsafe time does not exceed \(\epsilon T\). As explained in Section II-B, the confidence level \(1 - \beta\) can be made large without increasing the size \(M\) of samples significantly. This framework is useful in those situations where the system (1) is able to tolerate the exposure to a deteriorating agent for a limited amount of time. For example, let us consider a solar-powered autonomous vehicle. The regions without solar exposure are considered to be unsafe since the vehicle’s battery could be drained after a period of time. However, it would be
inefficient to plan a path for the vehicle completely avoiding all these shaded regions. Instead, a more reasonable requirement would be that the amount of time the vehicle spends in the shaded regions is small.

Remark 2: Our approach can also be used to characterize the case that there exists $t \in [0, T]$ such that $[z_{x_0}(t) - \xi^*, z_{x_0}(t) + \xi^*] \cap \text{Uns} \neq \emptyset$. For this case, we need to compute a value $\tau \geq 0$, which is larger than or equal to the amount of time such that $[z_{x_0}(t) - \xi^*, z_{x_0}(t) + \xi^*] \cap \text{Uns} \neq \emptyset$. Furthermore, we have that the amount of time the system (1) with the input $x_{0,i}$ spends inside the unsafe set $\text{Uns}$ does not exceed $\epsilon T + \tau$, with confidence of at least $1 - \beta$.

In the following, we use an example from a Van-der-Pol oscillator to understand the enhancement of our approach.

Example 1: Consider a system with $T = 10$, $x_{0,i} = (1.4, 2.3)\top$, and $\text{Uns} = \{y \in \mathbb{R} \mid y \geq 3\}$, whose internal dynamics are described by an ordinary differential equation which generally describes a Van-der-Pol oscillator [39]

$$
\begin{align*}
\frac{dx_1}{dt} &= x_2, \\
\frac{dx_2}{dt} &= (1 - x_1^2)x_2 - x_1.
\end{align*}
$$

We assume that the trajectory of the system (1) in this example describes the time evolution of the state $x_1$ in (10), i.e., $y(t) = b(x_{0,i}, t) = x_1(t)$ for $t \in [0, 10]$. The ground-truth trajectory $y_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R}$ is illustrated in Fig. 3. It is used to extract datum $(x_{0,i}, t, y_{1,i})_{j=1}^M$ and perform comparisons. The method of constructing the ground-truth trajectory is introduced in the beginning of Section II.

Let $\beta = 10^{-20}$ and $\epsilon = 0.01$. In this example, we use $M = 10811$ and a polynomial $w(e, x_{0,i}, t)$ of degree 6 over $t$ as a mathematical model to perform computations. Since $x_{0,i}$ is known, $w(e, x_{0,i}, t)$ is of the form $\sum_{i=0}^6 c_i t^i$. Note that the number $k+1$ of decision variables in (6) is 8 and consequently $M \geq 10811$ according to Theorem 2.

We obtain $\xi^* = 0.33$ via solving the linear program (6) with $U_c = U_\xi = 100$. Therefore, we have that with the confidence of at least $1 - 10^{-20}$

$$
y_{x_0,i}(t) \in [z_{x_0,i}(t) - 0.33, z_{x_0,i}(t) + 0.33]
$$

for all $t \in [0, 10]$ except at most a fraction of probability measure 0.01, where $z_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R}$ is the trajectory of the mathematical model $z(t) = w(e^*, x_{0,i}, t)$. We also take the time step $\Delta t = 10^{-5}$ and the corresponding states $(y_{x_0,i}(j\Delta t))_{j=1}^{10\top}$ on the ground-truth trajectory to verify the satisfiability of (11), i.e., whether $y_{x_0,i}(j\Delta t) \in [z_{x_0,i}(j\Delta t) - 0.33, z_{x_0,i}(j\Delta t) + 0.33]$ holds for $j \in \{0, 1, \ldots, 10^\top\}$. The satisfiability ratio is 100%.

Since $[z_{x_0,i}(t) - 0.33, z_{x_0,i}(t) + 0.33] \cap \text{Uns} = \emptyset$ for $t \in [0, 10]$, we have that the amount of time the system (1) with the input $(1.4, 2.3)\top$ spends inside the unsafe set $\text{Uns}$ does not exceed 0.1, with the confidence of at least $1 - 10^{-20}$.

2) Multiple Trajectories Verification: In Section III-B1, we considered one trajectory characterization of the system (1). In this section, we extend the method in Section III-B1 to multiple trajectories characterization. These trajectories are the ones of the system (1) with inputs $x_{0,1}, \ldots, x_{0,N}$.

This extension is straightforward. We just need to enrich the constraints in (6) by incorporating these discrete-time trajectories $(x_{0,1}, i, y_{1,i})_{j=1}^M, \ldots, (x_{0,N}, i, y_{N,i})_{j=1}^M$, consequently resulting in the following linear program:

$$
\begin{align*}
\min_{\xi} \xi \\
\text{s.t. for each } j = 1, \ldots, M \text{ and } i = 1, \ldots, N:
& w(e, x_{0,i}, t) - y_{1,i} \leq \xi \\
& y_{1,i} - w(e, x_{0,i}, t) \leq \xi \\
& -U_c \leq c_i \leq U_c, i = 1, \ldots, k \\
& 0 \leq \xi \leq U_\xi
\end{align*}
$$

(12)

where $U_c \in \mathbb{R}_{\geq 0}$ is a given upper bound for $c_i$, $i = 1, \ldots, k$, and $U_\xi \in \mathbb{R}_{\geq 0}$ is a given upper bound for $\xi$. Denote the optimal solution to (12) by $(e^{**}, \xi^{**})$.

We denote the trajectory of the mathematical model $z(t) = w(e^*, x_{0,i}, t)$ with the input $x_0$ by $z_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R}$. Similarly, we have the following theorem for the solution obtained via solving the linear program (12).

Theorem 3: Let $(e^{**}, \xi^{**})$ be an optimal solution to (12), $\epsilon \in (0, 1)$, $\beta \in (0, 1)$, and $\epsilon \geq \frac{2}{M} \left( \ln \frac{1}{\beta} + k + 1 \right)$.

(13)

Then for each input $x_{0,i}$, $i = 1, \ldots, N$, we have that with at least $1 - \beta$ confidence

$$
P_i(\{t \in [0, T] \mid |w(e^{**}, x_{0,i}, t) - b(x_{0,i}, t)| \leq \xi^{**}\}) \geq 1 - \epsilon.
$$

Proof: According to the scenario optimization in Section II-B, we have that with at least $1 - \beta$ confidence

$$
P_i(\{t \in [0, T] \mid \bigwedge_{i=1}^N |w(e^{**}, x_{0,i}, t) - b(x_{0,i}, t)| \leq \xi^{**}\}) \geq 1 - \epsilon.
$$

Since

$$
P_i(\{t \in [0, T] \mid |w(e^{**}, x_{0,i}, t) - b(x_{0,i}, t)| \leq \xi^{**}\}) \geq P_i(\{t \in [0, T] \mid \bigwedge_{j=1}^M |w(e^{**}, x_{0,i}, t) - b(x_{0,i}, t)| \leq \xi^{**}\})
$$

for $i \in \{1, \ldots, M\}$, the conclusion follows directly.

From Theorem 3, we have that for each trajectory $y_{x_0,i}(\cdot) : [0, T] \rightarrow \mathbb{R}$ of the system (1) with the input $x_{0,i}$, $i = 1, \ldots, N$, with confidence of at least $1 - \beta$

$$
y_{x_0,i}(t) \in [z_{x_0,i}(t) - \xi^{**}, z_{x_0,i}(t) + \xi^{**}]
$$
for all \( t \) in \([0, T]\) but at most a fraction of probability measure \( \epsilon \), i.e., with confidence of at least \( 1 - \beta \), each of the \( N \) trajectories of the system (1) deviates from the corresponding one of the mathematical model \( z(t) = w(e^{\epsilon \gamma}, x_0, t) \) by at most \( \xi^{**} \) for all \( t \in [0, T] \) but at most a fraction \( \epsilon \).

Consequently, the solution to Problem 2 for the system (1) with multiple trajectories is presented as follows. If \([z_{x_0}(t) - \xi^{**}, z_{x_0}(t) + \xi^{**}]\) does not intersect the unsafe set \( \text{Uns} \) for \( t \in [0, T], i \in [1, \ldots, N] \), we have that the amount of time the system (1) with the input \( x_{0,i} \) spends inside the unsafe set \( \text{Uns} \) does not exceed \( \epsilon T \), with confidence of at least \( 1 - \beta \).

It is worth remarking that the family of inputs \( (x_{0,i})_{i=1}^{N} \) here does not require to be extracted independently according to the probability distribution \( P_{x_0} \). They can be arbitrary \( N \) inputs of interest in the set \( \mathcal{X}_0 \).

**Example 2:** Let us take the system in Example 1 as an instance to illustrate the case of two trajectories verification. These two trajectories, which are presented in Fig. 4, respectively, describe the time evolution of the state \( x_i \) in (10) with two different inputs \( x_{0,1} = (1.25, 2.28)^{\top} \) and \( x_{0,2} = (1.55, 2.32)^{\top} \).

Let \( \beta = 10^{-20} \) and \( \epsilon = 0.01 \). In this example, we use \( M = 26211 \) and a polynomial \( w(e, x, t) \) of degree 6 as a mathematical model, which is input dependent and is linear in \( e \), to perform computations. The number \( k + 1 \) of decision variables in (6) is 85 and thus \( M \geq 26211 \) from Theorem 2.

We obtain \( \xi^{**} = 0.34 \) via solving the linear program (12) with \( U_c = U_{\xi} = 100 \). Thus, for each \( i = 1, 2 \), we have that with confidence of at least \( 1 - 10^{-20} \), \( y_{x_0}(t) \in [z_{x_0}(t) - 0.34, z_{x_0}(t) + 0.34] \) for all \( t \in [0, 10] \) except a small fraction 0.01, where \( z_{x_0}(\cdot) : [0, T] \to \mathbb{R} \) is the trajectory of the mathematical model \( z(t) = w(e^{\epsilon \gamma}, x_0, t) \). Like Example 1, within the Monte-Carlo testing framework, we take the time step \( \Delta t = 10^{-5} \) and the corresponding states \( (y_{x_0}(j\Delta t))_{j=0}^{10^6} \) on the ground-truth trajectory with the input \( x_{0,i} \) to verify whether \( y_{x_0}(j\Delta t) \in [z_{x_0}(j\Delta t) - 0.34, z_{x_0}(j\Delta t) + 0.34] \) for \( j \in \{0, 1, \ldots, 10^6\} \), where \( i = 1, 2 \). The satisfiability ratio is 100% for both of these two trajectories.

Since \([z_{x_0}(t) - 0.34, z_{x_0}(t) + 0.34] \cap \text{Uns} = \emptyset \) for \( t \in [0, 10] \) and \( i = 1, 2 \), we have that the amount of time of the system (1) with each of the two inputs \( x_{0,1} = (1.25, 2.28)^{\top} \) and \( x_{0,2} = (1.55, 2.32)^{\top} \) spends inside the unsafe set \( \text{Uns} \) does not exceed 0.1, with confidence of at least \( 1 - 10^{-20} \).

3) All Trajectories Verification: In this section, we further extend the method in Section III-B2 for multiple trajectories verification to all trajectories verification of the system (1) with the input set \( \mathcal{X}_0 \). Unlike in Section III-B2, the family of inputs \( (x_i)_{i=1}^{N} \) in this situation should be extracted independently according to the probability distribution \( P_{x_0} \).

**Theorem 4:** Let \( (e^{**}, \xi^{**}) \) be an optimal solution to (12), \( e_1 \in (0, 1) \), \( \beta_1 \in (0, 1) \), \( e_2 \in (0, 1) \), \( \beta_2 \in (0, 1) \), and

\[
\begin{align*}
\epsilon_1 & \geq \frac{2}{M} \ln \left( \frac{1}{\beta_1} + k + 1 \right) \\
\epsilon_2 & \geq \frac{2}{N} \ln \left( \frac{1}{\beta_2} + k + 1 \right).
\end{align*}
\]

Then, we have that with at least \( 1 - \beta_2 \) confidence, \( P_{x_0}(x_0 | x_0 \in \mathcal{X}) \geq 1 - \epsilon_2 \), where \( \mathcal{X} = \left\{ x_0 \in \mathcal{X}_0 \left| \left( t \in [0, T], \left| w(e^{**}, x_0, t) - b(x_0, t) \right| \leq \xi^{**} \right) \right. \right\} \).

**Proof:** Let us fix the time instances \( t_1, \ldots, t_M \) first, we have that with confidence of at least \( 1 - \beta_2 \)

\[
P_{x_0}\left( \left\{ x_0 \in \mathcal{X}_0 \left| \bigwedge_{j=1}^{M} \left| w(e^{**}, x_0, t_j) - b(x_0, t_j) \right| \leq \xi^{**} \right. \right. \right) \geq 1 - \epsilon_2.
\]

Let \( \tilde{\mathcal{X}}_0 = \left\{ x_0 \in \mathcal{X}_0 \left| \bigwedge_{j=1}^{M} \left| w(e^{**}, x_0, t_j) - b(x_0, t_j) \right| \leq \xi^{**} \right. \right\} \). Obviously, \( x_0, i \in \tilde{\mathcal{X}}_0, i = 1, \ldots, N \). For \( x_0 \in \tilde{\mathcal{X}}_0 \), we can add the constraints involving \( x_0 \) to the linear program (12) and obtain the following linear program:

\[
\min_{e_1} \xi^{**}
\text{s.t. for each } j = 1, \ldots, M \text{ and } i = 1, \ldots, N:
\begin{align*}
w(e, x_0, t_j) - y_{i,j} & \leq \xi^{**} \\
y_{i,j} - w(e, x_0, t_j) & \leq \xi^{**} \\
w(e, x_0, t_j) - b(x_0, t_j) & \leq \xi^{**} \\
-b(x_0, t_j) - w(e, x_0, t_j) & \leq \xi^{**} \\
-\xi^{**} & \leq \xi^{**} \\
0 & \leq \xi^{**} \leq U_{\xi^{**}}
\end{align*}
\]

Obviously, \( (e^{**}, \xi^{**}) \) is also an optimal solution to (16). Since the time instances \( t_1, \ldots, t_M \) are also extracted independently according to the distribution \( P_t \), Theorem 3 indicates that with confidence of at least \( 1 - \beta_1 \)

\[
P_t(t \in [0, T], \left| w(e^{**}, x_0, t) - b(x_0, t) \right| \leq \xi^{**}) \geq 1 - \epsilon_1
\]
for \( x_0 \in \tilde{X}_0 \). Thus, we have \( \tilde{X}_0 \subseteq \mathcal{X} \) and consequently the conclusion follows.

From Theorem 4, we have that with confidence of at least \( 1 - \beta_z \), the probability measure of the set \( \mathcal{X} \) is larger than \( 1 - \epsilon_2 \). The set \( \mathcal{X} \) is a set of inputs such that the trajectory of the system (1) with each of them does not deviate from the corresponding one of the model \( z(t) = w(\epsilon^*, \cdot , t) : \mathbb{R}^n \times [0,T] \rightarrow \mathbb{R} \) by \( \xi^* \) for all \( t \in [0,T] \) but at most a fraction \( \epsilon_1 \).

Thus, the solution to Problem 2 for the system (1) with all trajectories originating from the set \( X_0 \) is presented as follows. If \([z_{0}(t) - \xi^*, x_{0}(\cdot) + \xi^*] \cap \text{Uns} = \emptyset \) for \( x_0 \in X_0 \) and \( t \in [0,T] \), we have that with confidence of at least \( 1 - \beta_z \) the probability measure of inputs in \( X_0 \) such that the amount of time the system (1) with each of them spends inside \( \text{Uns} \) does not exceed \( \epsilon_1 T \) with confidence of at least \( 1 - \beta_1 \), is larger than \( 1 - \epsilon_2 \).

Although the size of the linear program (12) for computing PAC models does not depend on the dimension of the system (1), it heavily depends on \( \epsilon_1, \beta_1, \epsilon_2, \beta_2 \), and the number of unknown parameters in a prespecified PAC model template according to inequalities (14) and (15) in Theorem 4.

**Example 3:** Let us take the system in Example 1 again as an instance to illustrate the case of all trajectories characterization. The input set is assumed to be \( X_0 = [1.25, 1.55] \times [2.28, 2.32] \).

Let \( \beta_1 = 10^{-10}, \epsilon_1 = 0.3, \beta_2 = 10^{-10}, \) and \( \epsilon_2 = 0.5 \). In this example, we use \( M = 207, N = 125 \), and a polynomial \( w(\epsilon, t) \) of degree 6 as a mathematical model, which is input independent and is linear in \( \epsilon \), to perform computations. The number \( k + 1 \) of decision variables in (12) is 8 and consequently \( M \geq 207 \) and \( N \geq 125 \) according to Theorem 4. The computation time for solving the resulting linear program is 150.32 s. The reason that an input-independent model is used is to reduce the number of decision variables in (12), which further results in reduction of the size of extracted samples according to inequalities (14) and (15) and thus reduction of the size of the linear program (12). These computations were performed on an i7-7500U 2.70-GHz CPU with 32-GB RAM running Windows 10.

We obtain \( \xi^* = 0.38 \) via solving the linear program (12) with \( U_\epsilon = U_\xi = 100 \). Therefore, with confidence of at least \( 1 - 10^{-10} \), the probability measure of inputs in \( X_0 \) such that with confidence of at least \( 1 - 10^{-10} \)

\[
y_{x_0}(t) \in [z_{x_0}(t) - 0.38, z_{x_0}(t) + 0.38]
\]

for all \( t \in [0,10] \) but at most a fraction 0.3, is larger than 0.5, where \( z_{x_0}(\cdot) : [0,T] \rightarrow \mathbb{R} \) is the trajectory of the mathematical model \( z(t) = w(\epsilon^*, t) \). The reachability analysis is illustrated in Fig. 5. Within the Monte-Carlo testing framework, we extract \( 10^4 \) inputs \( (x_{0,i},t_{i})^{10^4}_{i=1} \) from \( X_0 \) independently according to the probability distribution \( P_{x_0} \) and then obtain their corresponding ground-truth trajectories for validating the above conclusion. This is illustrated in Fig. 6. Like Example 1, we take the time step \( \Delta t = 10^{-5} \) and the states \( (y_{x_0,i}(j\Delta t))^{10^6}_{j=0} \) on the ground-truth trajectory with the input \( x_{0,i} \) to verify the satisfiability of (17), where \( i = 1, \ldots, 10^4 \). The satisfiability

\[
\begin{align*}
&\hat{x}_1(t) = 3x_3(t) - x_1(t)x_4(t), \\
&\hat{x}_2(t) = x_4(t) - x_2(t)x_4(t), \\
&\hat{x}_3(t) = x_1(t)x_6(t) - 3x_3(t), \\
&\hat{x}_4(t) = x_2(t)x_6(t) - x_4(t), \\
&\hat{x}_5(t) = 3x_3(t) + 5x_1(t) - x_5(t), \\
&\hat{x}_6(t) = 5x_5(t) + 3x_3(t) + x_4(t) \\
&\quad - x_6(t)x_1(t) + x_2(t) + 2x_9(t) + 1, \\
&\hat{x}_7(t) = 5x_4(t) + x_2(t) - 0.5x_7(t), \\
&\hat{x}_8(t) = 5x_7(t) - 2x_5(t)x_6(t) + x_9(t) - 0.2x_8(t), \\
&\hat{x}_9(t) = 2x_6(t)x_8(t) - x_9(t).
\end{align*}
\]

Let \( \epsilon_1 = 0.2, \beta_1 = 10^{-10}, \epsilon_2 = 0.3, \) and \( \beta_2 = 10^{-10} \). In this example, we compute two polynomial models of degrees 2 and 5 to illustrate our method.

1) We use \( M = 271, N = 181 \), and a polynomial \( w(\epsilon, t) \) of degree 2 as a mathematical model, which is input

\[
\begin{align*}
&\hat{x}_1(t) = 3x_3(t) - x_1(t)x_4(t), \\
&\hat{x}_2(t) = x_4(t) - x_2(t)x_4(t), \\
&\hat{x}_3(t) = x_1(t)x_6(t) - 3x_3(t), \\
&\hat{x}_4(t) = x_2(t)x_6(t) - x_4(t), \\
&\hat{x}_5(t) = 3x_3(t) + 5x_1(t) - x_5(t), \\
&\hat{x}_6(t) = 5x_5(t) + 3x_3(t) + x_4(t) \\
&\quad - x_6(t)x_1(t) + x_2(t) + 2x_9(t) + 1, \\
&\hat{x}_7(t) = 5x_4(t) + x_2(t) - 0.5x_7(t), \\
&\hat{x}_8(t) = 5x_7(t) - 2x_5(t)x_6(t) + x_9(t) - 0.2x_8(t), \\
&\hat{x}_9(t) = 2x_6(t)x_8(t) - x_9(t).
\end{align*}
\]
Fig. 7. Illustration of trajectories reachability for Example 4 with the polynomial PAC model of degree 2. The green curves denote the extracted 181 trajectories. The red curves denote \( w(e^{**}, \cdot) + \xi^{**} : [0, T] \rightarrow \mathbb{R} \) and \( w(e^{**}, \cdot) - \xi^{**} : [0, T] \rightarrow \mathbb{R} \), respectively.

For all \( t \in [0, 10] \) but at most a fraction 0.2, is larger than 0.7, where \( z_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R} \) is the trajectory of the model \( z(t) = w(e^{**}, t) \). The reachability analysis is illustrated in Fig. 7. Like Example 3, within the Monte-Carlo framework, we also extract \( 10^5 \) inputs \( (x'_i, 0)_{i=1}^{10^5} \) to verify the conclusion, and obtain that the ratio of \( 10^4 \) inputs such that \( y_{x_0}(\cdot) (\Delta t) \in [z_{x_0}'(\cdot) (\Delta t) - 0.17, z_{x_0}'(\cdot) (\Delta t) + 0.17] \) for all \( i \in [0, 10] \) at most a fraction 0.05, is larger than 97.87\%, where \( \Delta t = 10^{-5} \). Since \( |z_{x_0}(t) - 0.17, z_{x_0}(t) + 0.17| \cap \mathbb{Uns} = \emptyset \) for \( t \in [0, 10] \) and \( x_0 \in X_0 \), we have that with at least \( 1 - 10^{-10} \) confidence, the probability measure of inputs in \( X_0 \) such that the amount of time the system (1) with each of them spends inside the unsafe set \( \mathbb{Uns} \) does not exceed 2 with confidence of at least \( 1 - 10^{-10} \), is larger than 0.7.

2) We use \( M = 301, N = 201 \), and a polynomial \( w(e, t) \) of degree 5 as a mathematical model, which is input independent and is linear in \( e \), to perform computations. Note that the number \( k + 1 \) of decision variables in (12) is 4 and consequently \( M \geq 271 \) and \( N \geq 181 \) according to Theorem 4. Via solving (12) with \( U_c = U_t = 100 \), we obtain \( \xi^{**} = 0.17 \). The computation time is 167.43 s. Therefore, according to Theorem 4, we conclude that with at least \( 1 - 10^{-10} \) confidence, the probability measure of inputs in \( X_0 \) such that with confidence of at least \( 1 - 10^{-10} \)

\[
y_{x_0}(t) \in [z_{x_0}(t) - 0.12, z_{x_0}(t) + 0.12]
\]

for all \( t \in [0, 10] \) but at most a fraction 0.2, is larger than 0.7, where \( z_{x_0}(\cdot) : [0, T] \rightarrow \mathbb{R} \) is the trajectory of the model \( z(t) = w(e^{**}, t) \). The reachability analysis is illustrated in Fig. 8. Within the Monte-Carlo framework, we use the \( 10^5 \) inputs \( (x'_i, 0)_{i=1}^{10^5} \) in the first case to verify the conclusion, and obtain that the ratio of \( 10^4 \) inputs such that \( y_{x_0}(\cdot) (\Delta t) \in [z_{x_0}'(\cdot) (\Delta t) - 0.12, z_{x_0}'(\cdot) (\Delta t) + 0.12] \) for all \( i \in [0, 10^5] \) but at most a fraction 0.05, is larger than 98.56\%, where \( \Delta t = 10^{-5} \). Similarly, due to the fact that \( [z_{x_0}(t) - 0.12, z_{x_0}(t) + 0.12] \cap \mathbb{Uns} = \emptyset \) for \( t \in [0, 10] \) and \( x_0 \in X_0 \), we have that with at least \( 1 - 10^{-10} \) confidence, the probability measure of inputs in \( X_0 \) such that the amount of time the system (1) with each of them spends inside the unsafe set \( \mathbb{Uns} \) does not exceed 2 with confidence of at least \( 1 - 10^{-10} \), is larger than 0.7. From the comparison results illustrated in Fig. 9 for the above two cases with the same PAC guarantees, i.e., \( \epsilon_1, \epsilon_2, \beta_1, \) and \( \beta_2 \) are the same, we observe that polynomial models of higher degree could describe the internal dynamics of the system (1) more exactly, but with more computation time.

Example 5: To demonstrate the applicability of our approach to higher dimensional systems, we consider a scalable system of the form (1) with \( T = 2, X_0 = [0.5, 0.6]^{101} \), and \( \mathbb{Uns} = \{ y \in \mathbb{R} \mid y \geq 3.0 \} \), describing the time evolution of the state \( x_l \) in an ordinary differential [33]

\[
\begin{align*}
\dot{x}_1(t) &= 1 + \frac{1}{2} (x_1(t) + x_2(t)) \\
\dot{x}_2(t) &= x_3(t), \quad \dot{x}_3(t) = -10 \sin x_2(t) - x_2(t) \\
\vdots \\
\dot{x}_l(t) &= x_{l-1}(t), \quad \dot{x}_{l+1}(t) = -10 \sin x_l(t) - x_l(t)
\end{align*}
\]

where \( l = 50 \).

Let \( \epsilon_1 = 0.2, \beta_1 = 10^{-10}, \epsilon_2 = 0.2, \) and \( \beta_2 = 10^{-10} \). In this example, we compute two polynomial models of degrees 2 and 4 to illustrate our method.

1) We use \( M = 271, N = 271 \), and a polynomial \( w(e, t) \) of degree 2 as a mathematical model, which is input independent, to perform computations. Note that the
number $k + 1$ of decision variables in (12) is 4 and consequently $M \geq 271$ and $N \geq 271$ according to Theorem 4. Via solving (12) with $U_c = U_\xi = 100$, we obtain that $\xi^* = 0.36$. The computation time is 398.23 s. According to Theorem 4, we have that with at least $1 - 10^{-10}$ confidence, the probability measure of inputs in $X_0$ such that with confidence of at least $1 - 10^{-10}$

$$y_{x_0}(t) \in [z_{x_0}(t) - 0.36, z_{x_0}(t) + 0.36]$$

for all $t \in [0, 2]$ but at most a fraction 0.2, is larger than 0.8, where $z_{x_0}(\cdot) : [0, T] \mapsto \mathbb{R}$ is the trajectory of the mathematical model $z(t) = w(e^{\xi*t}, t)$. The reachability analysis is illustrated in Fig. 10. Like Example 4, within the Monte-Carlo testing framework, we also extract $10^4$ inputs $(x_1^{(i)}, \omega_1^{(i)}), i = 1, \ldots, 10^4$ to verify the above conclusion, and obtain that the ratio of $10^4$ inputs such that $y_{x_1}(j\Delta t) \in [z_{x_1}(j\Delta t) - 0.36, z_{x_1}(j\Delta t) + 0.36]$ for all $j \in [0, \ldots, 10^4]$ is equal to 98.07%, where $\Delta t = (2/10^5)$ and $i = 1, \ldots, 10^4$. Since $[z_{x_0}(t) - 0.36, z_{x_0}(t) + 0.36] \cap \mathbb{R} \neq \emptyset$ for $t \in [0, 2]$ and $x_0 \in X_0$, we have that with at least $1 - 10^{-10}$ confidence, the probability measure of inputs in $X_0$ such that the amount of time the system (1) with each of them spends inside the unsafe set $\mathbb{R} \neq \emptyset$ does not exceed 0.4 with at least $1 - 10^{-10}$ confidence, is larger than 0.8.

2) We use $M = 291$, $N = 291$, and a polynomial $w(e, t)$ of degree 4 as a mathematical model, which is input independent, to perform computations. Note that the number $k + 1$ of decision variables in (12) is 6 and consequently $M \geq 291$ and $N \geq 291$ according to Theorem 4.Via solving (12) with $U_c = U_\xi = 100$, we obtain that $\xi^* = 0.12$. The computation time is 502.02 s. According to Theorem 4, we have that with at least $1 - 10^{-10}$ confidence, the probability measure of inputs in $X_0$ such that with confidence of at least $1 - 10^{-10}$

$$y_{x_0}(t) \in [z_{x_0}(t) - 0.12, z_{x_0}(t) + 0.12]$$

for all $t \in [0, 2]$ but at most a fraction 0.2, is larger than 0.8, where $z_{x_0}(\cdot) : [0, T] \mapsto \mathbb{R}$ is the trajectory of the mathematical model $z(t) = w(e^{\xi*t}, t)$. The reachability analysis is illustrated in Fig. 11. Also, within the Monte-Carlo testing framework, we use $10^4$ inputs $(x_1^{(i)}, \omega_1^{(i)}), i = 1, \ldots, 10^4$ in the first case to verify the above conclusion and obtain that the ratio of $10^4$ inputs such that $y_{x_1}(j\Delta t) \in [z_{x_1}(j\Delta t) - 0.12, z_{x_1}(j\Delta t) + 0.12]$, for all $j \in \{0, \ldots, 10^4\}$ is equal to 1, where $\Delta t = (2/10^5)$ and $i = 1, \ldots, 10^4$. Similar to the first case, we have that with at least $1 - 10^{-10}$ confidence, the probability measure of inputs in $X_0$ such that the amount of time the system (1) with each of them spends inside the unsafe set $\mathbb{R} \neq \emptyset$ does not exceed 0.4 with confidence of at least $1 - 10^{-10}$, is larger than 0.8. Like Example 4, by comparing the results in Fig. 12 for the above two cases with the same PAC guarantees, i.e., $\epsilon_1, \epsilon_2, \beta_1,$ and $\beta_2$ are the same, we also obtain that polynomial models of higher degree could capture the internal dynamics of the system (1) more exactly, but with more computation time.

**Example 6:** In this example, we show a strategy to overcome the issue of solving large-scale linear programs based on a black-box system of the form (1) which describes the time evolution of the state $x_1$ in the 2-D delay differential equation

$$\begin{align*}
\dot{x}_1(t) &= a x_1(t) \left(1 - \frac{x_1(t)}{m} \right) + b x_2(t - \tau) x_1(t - \tau) \\
\dot{x}_2(t) &= c x_2(t) + d x_1(t - \tau) x_2(t - \tau)
\end{align*}$$

where $\tau = 0.1$, $a = 0.25$, $m = 200$, $b = -0.01$, $c = -1.00$, and $d = 0.01$. The delay differential equation was a model for predator-prey populations.

Assume that $T = 10$, the initial condition $x(t)$ over $t \in [-0.1, 0]$ is a constant vector falling within $X_0 = \{(x_1, x_2) \mid (x_1 + 5)^2 + (x_2 + 5)^2 \leq 1 \}$ and $\mathbb{R} \neq \emptyset = \{y \mid y \geq 40\}$. Let $\epsilon_1 = 0.1, \beta_1 = 10^{-10}, \epsilon_2 = 0.1, \beta_2 = 10^{-10}$. In this example, we first use input-dependent polynomial models of degree 4 to illustrate this strategy, and then use input-independent polynomial models of degree 4 to illustrate it.

1) **Input-Dependent Models:** If a generic polynomial input-dependent model template of degree 4, which is formed
by choosing all monomials of degree up to 4 as the basis polynomials, is employed, the number \(k+1\) of decision variables in (12) is 36 and consequently \(M \geq 1181\) and \(N \geq 1181\) according to Theorem 4. This leads to a large-scale linear program, producing a heavy computational burden. As a result, we did not obtain results within two hours via solving this large-scale linear program.

Our strategy for avoiding large-scale linear programs is as follows: a small family of datum is first employed to compute an initial estimate of the coefficients \(\hat{c}\) and leave the remaining ones unknown, reducing the number of decision variables in (12) and thus the size of the resulting linear program. In the experiment, we first solve the linear program (12) with \(M = 50\) and \(N = 50\) to obtain a model \(w^\prime(c^*, x, t)\) with the computation time of 1.82 s, and then use the computed \(w^\prime(c^*, x, t)\) to perform computations on the linear program (12) with \(M = N = 481\) and \(U_c = U_\xi = 100\). Note that the number \(k+1\) of decision variables in (12) becomes 1 in this setting and consequently \(M \geq 481\) and \(N \geq 481\) according to Theorem 4. Via solving (12) with \(U_c = U_\xi = 100\), we obtain that \(\xi^{**} = 1.49\) with the computation time of 268.67 s. The reachability analysis is illustrated in Fig. 13. Therefore, according to Theorem 4, we conclude that with at least \(1 - 10^{-10}\) confidence, the probability measure of inputs in \(X_0\) such that with confidence of at least \(1 - 10^{-10}, y_0(t) \in [z_0(t) - 1.49, z_0(t) + 1.49]\) for all \(t \in [0, 10]\) but at most a fraction 0.1, is larger than 0.9, where \(z_0(\cdot):[0, T] \rightarrow \mathbb{R}\) is the trajectory of the mathematical model \(z(t) = w^\prime(c^*, x, t)\). Also, within the Monte-Carlo framework, we extract \(10^4\) inputs \((x_i^\prime)_{i=1}^{10^4}\) to verify the above conclusion and obtain that the ratio of \(10^4\) inputs such that \(y_0^\prime(j\Delta t) \in [z_0^\prime(j\Delta t) - 1.49, z_0^\prime(j\Delta t) + 1.49]\) for all \(j \in \{0, \ldots, 10^4\}\) is 100%, where \(\Delta t = 10^{-5}\). Since \([z_0(t) - 1.49, z_0(t) + 1.49] \cap \mathbb{U}_{\mathbb{U}} = \emptyset\) for \(t \in [0, 10]\) and \(x_0 \in X_0\), we have that with at least \(1 - 10^{-10}\) confidence, the probability measure of inputs in \(X_0\) such that the amount of time the system (1) with each of them spends inside \(\mathbb{U}_{\mathbb{U}}\) does not exceed 1 with a confidence of at least \(1 - 10^{-10}\), is larger than 0.9.

2) **Input-Independent Models:** If an input-independent polynomial template of degree 4 is used to perform computations, the number \(k+1\) of decision variables in (12) is 6 and consequently \(M \geq 581\) and \(N \geq 581\) according to Theorem 4. Via solving the linear program (12) with \(M = N = 581\) and \(U_c = U_\xi = 100\), we obtain \(\xi^{**} = 24.84\) with the computation time of 6634.51 s. The reachability analysis is illustrated in Fig. 14. We also adopt the strategy presented in the above case for reducing the computation cost. We first solve the linear program (12) with \(M = N = 50\) and \(U_c = U_\xi = 100\) to obtain a \(w^\prime(c^*, x, t)\) with the computation time of 1.65 s, and then use the computed \(w^\prime(c^*, x, t)\) to perform computations on the linear program (12) with \(M = N = 481\) and \(U_c = U_\xi = 100\).

Note that the number \(k+1\) of decision variables in (12) becomes 1 in this setting and consequently \(M \geq 481\) and \(N \geq 481\) according to Theorem 4. Via solving (12) with \(U_c = U_\xi = 100\), we obtain that \(\xi^{**} = 25.96\) with the computation time of 71.09 s. The reachability analysis is illustrated in Fig. 14 as well. The safety guarantee is the same with the case of using input-dependent models. Similarly, within the Monte-Carlo framework, we use the \(10^4\) inputs \((x_i^\prime)_{i=1}^{10^4}\) in the first case to verify the above conclusion and obtain that the ratio of \(10^4\) inputs such that \(y_0^\prime(j\Delta t) \in [z_0^\prime(j\Delta t) - 25.96, z_0^\prime(j\Delta t) + 25.96]\) for all \(j \in \{0, \ldots, 10^4\}\) is equal to 100%, where \(\Delta t = 10^{-5}\). Via comparing the results in Figs. 13 and 14 for the above two cases with the same PAC guarantees, i.e., \(\epsilon_1, \epsilon_2, \beta_1, \text{ and } \beta_2\) are the same, we conclude that input-dependent polynomial models could capture the internal dynamics of the system (1) more exactly than input-independent ones, but also with more computation cost.

**V. CONCLUSION**

In this article, we proposed a novel PAC model checking approach for finite-time safety verification of black-box continuous-time dynamical systems, which are represented by observed datum, within the framework of PAC learning. In this approach, a PAC model of the system was computed such that the time-evolving trajectories of the black-box dynamical system over finite-time horizons fall within the range of the PAC model plus a bounded interval with error probabilities and confidence levels, thus facilitating the formal characterization of the satisfiability of safety requirements. Both
the PAC model and the bounded interval were obtained via scenario optimization, which finally boils down to a linear program. Three examples demonstrated the performance of our approach.

In the future, we would extend our method to safety verification of black-box systems, whose internal mechanisms are described by hybrid dynamical systems that exhibit both continuous and discrete dynamic behavior. Also, we would like to extend our method for the safety verification of black-box systems with noise measurements and inputs.

REFERENCES

[1] B. K. Aichernig and M. Tappler, “Probabilistic black-box reachability checking (extended version),” in Proc. FME, 2014, pp. 98–114.
[2] G. C. Calafiore and M. C. Campi, “The scenario approach to robust control design,” IEEE Trans. Autom. Control, vol. 51, no. 5, pp. 742–753, May 2006.
[3] M. C. Campi, S. Garatti, and M. Prandini, “The scenario approach for systems and control design,” Ann. Rev. Control, vol. 33, no. 2, pp. 149–157, 2009.
[4] T. Brázdil et al., “Verification of Markov decision processes using learning algorithms,” in Proc. ATVA, 2014, pp. 223–235.
[5] E. M. Clarke, P. Zuliani, “A Bayesian approach to model checking biological systems,” in Proc. CMSB, 2009, pp. 218–234.
[6] J.-P. Katoen, I. S. Zapreva, E. M. Hahn, H. Hermanns, and D. N. Jansen, “The ins and outs of the probabilistic model checker MRMC,” Perform. Eval., vol. 68, no. 2, pp. 90–104, 2011.
[7] B. Xue, M. Fränzle, H. Zhao, N. Zhan, and A. Easwaran, “Over- and under-approximating reach sets for perturbed delay differential equations,” in Proc. CAV, 2020, pp. 1–13.
[8] B. Xue, Q. Wang, S. Feng, and N. Zhan, “Safe inputs approximation for black-box systems,” in Proc. IEEE ICCECCS, 2019, pp. 180–189.
[9] B. Xue, Y. Liu, L. Ma, X. Zhang, M. Sun, and X. Xie, “Safe inputs approximation for black-box systems,” in Proc. IEEE ICCECCS, 2019, pp. 180–189.
[10] B. Xue, Q. Wang, S. Feng, and N. Zhan, “Over- and under-approximating reach sets for perturbed delay differential equations,” IEEE Trans. Autom. Control, early access, Mar. 3, 2020, doi: 10.1109/TAC.2020.2977993.
[11] B. Xue, Y. Liu, L. Ma, X. Zhang, M. Sun, and X. Xie, “Safe inputs approximation for black-box systems,” in Proc. IEEE ICCECCS, 2019, pp. 180–189.
[12] B. Xue, Q. Wang, S. Feng, and N. Zhan, “Over- and under-approximating reach sets for perturbed delay differential equations,” IEEE Trans. Autom. Control, early access, Mar. 3, 2020, doi: 10.1109/TAC.2020.2977993.
[13] B. Xue, Y. Liu, L. Ma, X. Zhang, M. Sun, and X. Xie, “Safe inputs approximation for black-box systems,” in Proc. IEEE ICCECCS, 2019, pp. 180–189.