Automata Theory Meets Barrier Certificates: Temporal Logic Verification of Nonlinear Systems

Article *in* IEEE Transactions on Automatic Control · March 2014

DOI: 10.1109/TAC.2015.2511722 · Source: arXiv

3 authors, including:

Andrew Lamperski
University of Minnesota
43 PUBLICATIONS **444** CITATIONS

Automata Theory Meets Barrier Certificates: Temporal Logic Verification of Nonlinear Systems

Tichakorn Wongpiromsarn* Ufuk Topcu† Andrew Lamperski‡

***Abstract*— We consider temporal logic verification of (possibly nonlinear) dynamical systems evolving over continuous state spaces. Our approach combines automata-based verification and the use of so-called barrier certificates. Automata-based verification allows the decomposition the verification task into a finite collection of simpler constraints over the continuous state space. The satisfaction of these constraints in turn can be (potentially conservatively) proved by appropriately constructed barrier certificates. As a result, our approach, together with optimization-based search for barrier certificates, allows computational verification of dynamical systems against temporal logic properties while avoiding explicit abstractions of the dynamics as commonly done in literature.**

I. Introduction

We propose a sound but incomplete method for the computational verification of specifications expressed in temporal logic against the behavior of dynamical systems evolving over (potentially partially) continuous state spaces. This new method merges ideas from automata-based model checking with those from control theory including so-called barrier certificates and optimization-based search for such certificates. More specifically, we consider linear temporal logic (excluding the "next" operator) formulas over atomic propositions that capture (sub)set memberships over the continuous state space. Under mild assumptions, the properties of the trajectories, which are salient for the verification, of the system can be characterized by infinite sequences (we call them *traces*) that track the atomic propositions satisfied along the corresponding trajectories (i.e., the subsets visited along the trajectory). Then, an automaton representation of the negation of the temporal logic formula guides a decomposition of the verification task into a finite collection of simpler constraints over the continuous state space. The satisfaction of these constraints in turn can be (potentially conservatively) proved by appropriately constructed barrier certificates.

Verification of dynamical systems against rich temporal logic specification has attracted considerable attention. A widely explored approach is based on proving (or disproving) (e.g., by using model checking [1], [2]) the specification using finite-state abstractions of the underlying dynamics [3], [4]. The consistency of the satisfaction of the specifications by the dynamical system and its finite-state abstractions is established through simulation and bi-simulation relations [5] or approximately through approximate bi-simulation relations [6]. In general, these existing approaches are not complete, except for certain simple dynamics [7]. In addition, the abstract finite state systems are often large, leading to the state explosion problem.

The method we propose avoids explicit abstractions of the dynamics. On the other hand, the automaton representation of the specification may be interpreted as a "minimal" finite-state abstraction required for verification. The details due to the dynamics ignored in this abstraction are then accounted for by the barrier certificates only to the level of fidelity and locally over the regions of the continuous state space dictated by the dynamics. However, similar to existing approaches for verifying nonlinear systems against temporal logic specifications, our approach is also not complete.

Not as rich as linear temporal logic but barrier certificates were originally considered to prove the satisfaction of temporal constraints, e.g., safety, reachability, and eventuality, for dynamical systems [8], [9]. Reference [9] also demonstrated the use of multiple and/or more sophisticated[1] barrier certificates for verifying properties beyond the basic ones mentioned above. Furthermore, one can imagine that it may be possible to look for increasingly complicated barrier certificates to verify arbitrary linear temporal logic specifications. The main contribution of this paper is to partly formalize such imagination by systematically constructing a collection of barrier certificates which all together witness the satisfaction of arbitrary linear temporal logic specifications.

The method developed in this paper is in principle applicable to a broad family of dynamical systems as long as certain, relatively mild smoothness conditions hold. In the presentation we consider continuous vector fields for simplicity. The step, which practically determines the applicability, of the proposed procedure is the computational search for barrier certificates. In this step, we focus on polynomial vector fields and resort to a combination of generalizations of the S-procedure [10], [11] and sum-of-squares relaxations for global polynomial optimization [10]. These techniques are relatively standard now in controls and have been used in other work on quantitative analysis of nonlinear and hybrid systems [9], [12], [13], [14], [15].

The rest of the paper is organized as follows: We begin with some notation and preliminaries needed in the rest of the paper. The problem formulation in section III is

* Thailand Center of Excellence for Life Sciences, Thailand (tichakorn@tcels.or.th)

† , University of Pennsylvania, Philadelphia, PA (utopcu@seas.upenn.edu)

‡ , University of Cambridge, United Kingdom (a.lamperski@eng.cam.ac.uk)

[1] Informally in terms of the conditions that need to be satisfied by the corresponding barrier certificates.

followed by the automata-theoretic notions in section IV which characterize the verification as checking properties of potentially infinitely many run fragments. Section V reduces this checking to a finite set of representative run fragments. Section VI discusses the role of the barrier certificates. Section VII puts the pieces introduced in the earlier sections together and gives a pseudo-algorithm as well as pointers to some of the computational tools required to implement the algorithm. The critique in section VIII is followed by an application of the method to an example, which is also used as a running example throughout the paper.

II. Preliminaries

In this section, we define the formalism used in the paper to describe systems and their desired properties. Given a set X, we let 2^X and $|X|$ denote the powerset and the cardinality of X, respectively, and let X^*, X^+ and X^ω denote the set of finite, nonempty finite and infinite strings of X. For finite strings σ_1 and σ_2, let $\sigma_1\sigma_2$ denote a string obtained by concatenating σ_1 and σ_2, σ_1^* and σ_1^+ denote a finite string and a nonempty finite string, respectively, obtained by concatenating σ_1 finitely many times and σ_1^ω denote an infinite string obtained by concatenating σ_1 infinitely many times. Given a finite string $\sigma = a_0a_1\dots a_m$ where $m \in \mathbb{N}$ or an infinite string $\sigma = a_0a_1\dots$, a *substring* of σ is any finite string $a_ia_{i+1}\dots a_{i+k}$ where $i, k \geq 0$ and $i+k \leq m$ if σ is finite. Finally, for any $\mathcal{Y} \subseteq \mathbb{R}^n$ where $n \in \mathbb{N}$, we let $\overline{\mathcal{Y}}$ be the closure of $\mathcal{Y}$ in $\mathbb{R}^n$.

Consider a dynamical system $\mathbb{D}$ whose state $x \in \mathcal{X} \subseteq \mathbb{R}^n$, $n \in \mathbb{N}$ evolves according to the differential equation

$$\dot{x}(t) = f(x(t)). \tag{1}$$

Let (by slight abuse of notation) $x : \mathbb{R}_{\geq 0} \to \mathcal{X}$ also represent a trajectory of the system, i.e., a solution of (1). We assume that the vector field f is continuous to ensure that its solution x is piecewise continuously differentiable.

A. Barrier Certificates

We are interested in verifying the system in (1) against a broad class of properties (whose definition and semantics will be introduced later) that roughly speaking temporally and logically constrain the evolution of the system. A building block in the subsequent development is the use of the so-called barrier certificates which, in recent literature [9], were utilized to verify safety, reachability, and other simple specifications that can essentially be interpreted as instances of the specification language considered in this paper. We now introduce a barrier certificate-type result as a prelude. This result will later be invoked in section VI.

Lemma 1: Let $\mathcal{Y}, \mathcal{Y}_0, \mathcal{Y}_1 \subseteq \mathcal{X}$. Suppose there exists a differentiable function $B : \mathcal{X} \to \mathbb{R}$ that satisfies the following conditions:

$$B(x) \leq 0 \quad \forall x \in \mathcal{Y}_0, \tag{2}$$

$$B(x) > 0 \quad \forall x \in \overline{\mathcal{Y}_1}, \tag{3}$$

$$\frac{\partial B}{\partial x}(x) f(x) \leq 0 \quad \forall x \in \overline{\mathcal{Y}} \setminus \overline{\mathcal{Y}_1}. \tag{4}$$

Then, any trajectory of $\mathbb{D}$ that starts in $\mathcal{Y}_0$ cannot reach $\mathcal{Y}_1$ without leaving $\overline{\mathcal{Y}}$.

Proof: Consider a trajectory x of $\mathbb{D}$ that starts in $\mathcal{Y}_0$. Suppose x reaches $\mathcal{Y}_1$ without leaving $\overline{\mathcal{Y}}$. Then, there exists $T \in \mathbb{R}$ such that $x(T) \in \overline{\mathcal{Y}_1}$ and $x(t) \in \overline{\mathcal{Y}} \setminus \overline{\mathcal{Y}_1}$ for all $t \in [0, T)$. From conditions (2) and (3), we get that $B(x(0)) \leq 0$ and $B(x(T)) > 0$. In addition, condition (4) implies that $B(x(t)) \leq 0$ for all $t \in [0, T)$. From the continuity of x and B, we can conclude that $B(x(T)) \leq 0$, leading to a contradiction. ∎

Lemma 1 (potentially conservatively) translates a verification question (whether all solutions to (1) satisfy the specified temporal ordering between "visiting" $\mathcal{Y}_0$, $\mathcal{Y}_1$, and $\mathcal{Y}$) into search for a map that satisfies the algebraic conditions in (2)-(4).

Later, we develop a method for automatically deriving a finite collection of such algebraic conditions for the verification of temporal logic specifications which has been demonstrated to be an appropriate specification formalism for reasoning about various kinds of systems [16].

B. Linear Temporal Logic

We employ linear temporal logic without the next operator (LTL$_{\setminus\bigcirc}$) to describe behaviors of continuous systems.[2] An LTL$_{\setminus\bigcirc}$ formula is built up from a set of *atomic propositions* and two kinds of operators: logical connectives and temporal modal operators. The logical connectives are those used in propositional logic: *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$) and *material implication* ($\Longrightarrow$). The temporal modal operators include *always* ($\Box$), *eventually* ($\Diamond$) and *until* ($\mathcal{U}$).

Definition 1: An *LTL$_{\setminus\bigcirc}$ formula* over a set Π of atomic propositions is inductively defined as follows:

(1) *True* is an LTL$_{\setminus\bigcirc}$ formula,
(2) any atomic proposition $p \in \Pi$ is an LTL$_{\setminus\bigcirc}$ formula, and
(3) given LTL$_{\setminus\bigcirc}$ formulas φ_1 and φ_2, the formulas $\neg\varphi_1$, $\varphi_1 \vee \varphi_2$, and $\varphi_1 \,\mathcal{U}\, \varphi_2$ are also LTL$_{\setminus\bigcirc}$ formulas.

Additional operators can be derived from the logical connectives $\vee$ and $\neg$ and the temporal modal operator $\mathcal{U}$. For example, $\varphi_1 \wedge \varphi_2 = \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \Longrightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$, $\Diamond\varphi = \mathit{True}\,\mathcal{U}\,\varphi$ and $\Box\varphi = \neg\Diamond\neg\varphi$.

LTL$_{\setminus\bigcirc}$ formulas are interpreted on infinite strings $\sigma = a_0a_1a_2\dots$ where $a_i \in 2^\Pi$ for all $i \geq 0$. Such infinite strings are referred to as *words*. The satisfaction relation is denoted by $\models$, i.e., for a word σ and an LTL$_{\setminus\bigcirc}$ formula φ, we write $\sigma \models \varphi$ if and only if σ satisfies φ and write $\sigma \not\models \varphi$ otherwise. The satisfaction relation is defined inductively as follows:

- $\sigma \models \mathit{True}$,
- for an atomic proposition $p \in \Pi$, $\sigma \models p$ if and only if $p \in a_0$,
- $\sigma \models \neg\varphi$ if and only if $\sigma \not\models \varphi$,
- $\sigma \models \varphi_1 \wedge \varphi_2$ if and only if $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$, and
- $\sigma \models \varphi_1 \,\mathcal{U}\, \varphi_2$ if and only if there exists $j \geq 0$ such that $a_ja_{j+1}\dots \models \varphi_2$ and for all i such all $0 \leq i < j$, $a_ia_{i+1}\dots \models \varphi_1$.

[2] Similar to [17], our choice of LTL$_{\setminus\bigcirc}$ over the widely used linear temporal logic that includes the next operator is motivated by our definition of the satisfaction of a formula with discrete time semantics by a continuous trajectory.

Given a proposition p, examples of widely used LTL$_{\setminus\bigcirc}$ formulas include a safety formula of the form $\Box p$ (read as "always p") and a reachability formula of the form $\Diamond p$ (read as "eventually p"). A word satisfies $\Box p$ if p remains invariantly true at all positions of the word whereas it satisfies $\Diamond p$ if p becomes true at least once in the word. By combining the temporal operators, we can express more complex properties. For example $\Box\Diamond p$ states that p holds infinitely often in the word.

Let φ be an LTL$_{\setminus\bigcirc}$ formula over Π. The linear-time property induced by φ is defined as $Words(\varphi) = \{\sigma \in (2^\Pi)^\omega \mid \sigma \models \varphi\}$.

C. Correctness of Dynamical Systems

As described in Section II-B, LTL$_{\setminus\bigcirc}$ formulas are interpreted on infinite strings. In this section, we show that the properties of trajectories of continuous systems can be characterized by such infinite strings, allowing LTL$_{\setminus\bigcirc}$ formulas to be interpreted over continuous trajectories.

The behavior of the system is formalized by a set Π of atomic propositions where each atomic proposition $p \in \Pi$ corresponds to a region of interest $[\![p]\!] \subseteq \mathcal{X}$. Following [17], [18], we define a trace of a trajectory to be the sequence of sets of propositions satisfied along the trajectory. Specifically, for each $a \in 2^\Pi$, we define

$$[\![a]\!] = \begin{cases} \mathcal{X} \setminus \bigcup_{p\in\Pi} [\![p]\!] & \text{if } a = \emptyset \\ \bigcap_{p\in a} [\![p]\!] \setminus \bigcup_{p\in\Pi\setminus a} [\![p]\!] & \text{otherwise.} \end{cases} \quad (5)$$

According to Equation (5), $[\![\emptyset]\!]$ is the subset of $\mathcal{X}$ that does not satisfy any atomic proposition in Π whereas for any $a \in 2^\Pi$ such that $a \neq \emptyset$, $[\![a]\!]$ is the subset of $\mathcal{X}$ that satisfy all and only propositions in a.

Definition 2: An infinite sequence $\sigma_x = a_0 a_1 a_2 \ldots$ where $a_i \in 2^\Pi$ for all $i \in \mathbb{N}$ is a *trace* of a trajectory $x : \mathbb{R}_{\geq 0} \to \mathcal{X}$ of $\mathbb{D}$ if there exists an associated sequence $t_0 t_1 t_2 \ldots$ of time instances such that $t_0 = 0$, $t_k \to \infty$ as $k \to \infty$ and for each $i \in \mathbb{N}$, $t_i \in \mathbb{R}_{\geq 0}$ satisfies the following conditions:

(1) $t_i < t_{i+1}$,
(2) $x(t_i) \in [\![a_i]\!]$, and
(3) if $a_i \neq a_{i+1}$, then for some $t'_i \in [t_i, t_{i+1}]$, $x(t) \in [\![a_i]\!]$ for all $t \in (t_i, t'_i)$, $x(t) \in [\![a_{i+1}]\!]$ for all $t \in (t'_i, t_{i+1})$ and either $x(t'_i) \in [\![a_i]\!]$ or $x(t'_i) \in [\![a_{i+1}]\!]$.

See Figure 1 for a hypothetical example which explains the relation between a sample trajectory x and its trace σ_x. In this case, we have $x(t_0), x(t_2), x(t_4) \in \mathcal{X} \setminus \bigcup_{p\in\Pi} [\![p]\!]$, $x(t_1) \in [\![p_A]\!]$, $x(t_3) \in [\![p_B]\!]$ and $x(t_5) \in [\![p_C]\!]$. Hence, σ_x is given by $\sigma_x = \emptyset\{p_A\}\emptyset\{p_B\}\emptyset\{p_C\}\ldots$. Note that definition 2 is consistent with the definition of the word produced by a continuous trajectory in [17], [18] with slight differences. Specifically, the definition in [17] has an additional requirement that if for any $i \in \mathbb{N}$, $a_i = a_{i+1}$, then $[\![a_i]\!]$ has to be a "sink" for the trajectory, i.e., $x(t) \in [\![a_i]\!]$ for all $t \geq t_i$. Reference [18] requires the time sequence $t_0 t_1 t_2 \ldots$ in Defition 2 to be exactly the instances where the sets of propositions satisfied by the trajectory changes, i.e., $t_i = \inf\{t \mid t > t_{i-1}, x(t) \notin [\![a_{k-1}]\!]\}$ for all $i > 0$. We refer the reader to [18] for the discussion on the existence of traces of realistic trajectories (i.e., those of *finite variability*).

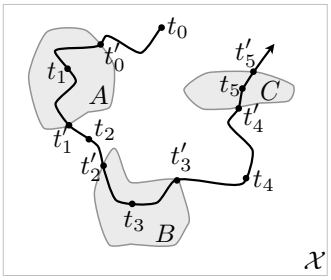

Fig. 1. A hypothetical example which explains the relation between a sample trajectory and its trace. As shown, $A, B, C \subset \mathcal{X}$. Let $\Pi = \{p_A, p_B, p_C\}$ where for each $S \in \{A, B, C\}$, $[\![p_S]\!] = S$. The trajectory x is represented by a solid curve starting at t_0. A time sequence $t_0 t_1 t_2 \ldots$ associated with a trace of x as well as the intermediate time instances $t'_0, t'_1, t'_2, \ldots$ satisfying condition 3 of Definition 2 are as shown.

An important feature of a trace is that it captures the instances where the characteristics of the states along the trajectory (as defined by a combination of atomic propositions in Π) change. That is, a trace of x characterizes the behavior of x according to the sequence of sets of propositions satisfied, which correspond to regions visited, along the trajectory. Finally, define $Trace(\mathbb{D}) = \{\sigma_x \in (2^\Pi)^\omega \mid$ there exists a trajectory x of $\mathbb{D}$ such that σ_x is a trace of $x\}$ to be the set of traces of trajectories of $\mathbb{D}$.

Next, we provide the definition of the satisfaction of an LTL$_{\setminus\bigcirc}$ formula by $\mathbb{D}$.

Definition 3: Given a trajectory x of a dynamical system $\mathbb{D}$ and an LTL$_{\setminus\bigcirc}$ formula φ over Π, we say that x *satisfies* φ if for each infinite string $\sigma_x \in (2^\Pi)^\omega$ that is a trace of x, $\sigma_x \models \varphi$, i.e., the behavior of x as captured by its trace is correct with respect to φ.

Definition 4: A dynamical system $\mathbb{D}$ *satisfies* φ if all trajectories of $\mathbb{D}$ satisfy φ, i.e., $Trace(\mathbb{D}) \subseteq Words(\varphi)$.

D. Automata Representation of LTL$_{\setminus\bigcirc}$ Formulas

There is a tight relationship between LTL$_{\setminus\bigcirc}$ and finite state automata that will be exploited in this paper.

Definition 5: A *non-deterministic Buchi automaton* (NBA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ where

- Q is a finite set of states,
- Σ is a finite set, called an alphabet,
- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation,
- $Q_0 \subseteq Q$ is a set of initial states, and
- $F \subseteq Q$ is a set of accepting (or final) states.

We use the relation notation, $q \xrightarrow{a} q'$, to denote $(q, a, q') \in \delta$.

Consider an NBA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$. Let π be a sequence of states of $\mathcal{A}$, i.e., $\pi = q_0 q_1 \ldots q_m$ for some $m \in \mathbb{N}$, if it is finite, and $\pi = q_0 q_1 \ldots$ where $q_i \in Q$ for all i, if it is infinite. We say that π is a *run fragment* of $\mathcal{A}$ if, for each i, there exists $a_i \in \Sigma$ such that $q_i \xrightarrow{a_i} q_{i+1}$. Hence, a finite run fragment $\pi = q_0 q_1 \ldots q_m$ of $\mathcal{A}$ generates a set $\mathcal{ST}(\pi) = \{a_0 a_1 \ldots a_{m-1} \in \Sigma^* \mid q_i \xrightarrow{a_i} q_{i+1}$ for all $i \in \{0, \ldots, m-1\}\}$ of finite strings and an

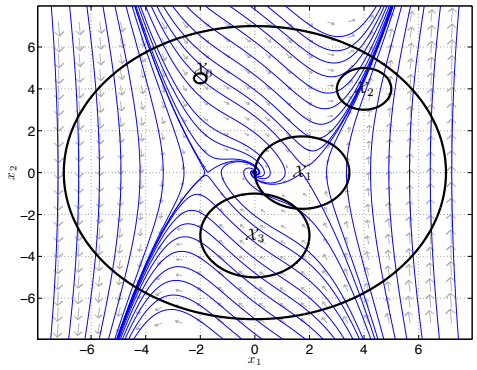

Fig. 2. Phase portrait of the dynamical system in (6), some representative trajectories (blue curves), and the sets $\mathcal{X}$, $\mathcal{X}_0, \ldots, \mathcal{X}_3$ defined in (7). Thick (black) curves are the boundaries of $\mathcal{X}$, $\mathcal{X}_0, \ldots, \mathcal{X}_3$ with the biggest circle being the boundary of $\mathcal{X}$.

infinite run fragment $\pi = q_0 q_1 \ldots$ generates a set $\mathcal{ST}(\pi) = \{a_0 a_1 \ldots \in \Sigma^\omega \mid q_i \xrightarrow{a_i} q_{i+1} \text{ for all } i\}$ of infinite strings. A *run* of $\mathcal{A}$ is an infinite run fragment $\pi = q_0 q_1 \ldots$ such that $q_0 \in Q_0$. Given an infinite string $\sigma = a_0 a_1 \ldots \in \Sigma^\omega$, a *run for* σ in $\mathcal{A}$ is an infinite sequence of states $\pi = q_0 q_1 \ldots$ such that $q_0 \in Q_0$ and $q_i \xrightarrow{a_i} q_{i+1}$ for all $i \geq 0$, i.e., $\sigma \in \mathcal{ST}(\pi)$. A run is *accepting* if there exist infinitely many $j \geq 0$ such that $q_j \in F$. A string $\sigma \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if there is an accepting run for σ in $\mathcal{A}$. The *language* accepted by $\mathcal{A}$, denoted by $\mathcal{L}_\omega(\mathcal{A})$, is the set of all accepted strings of $\mathcal{A}$.

It can be shown that for any LTL$_{\setminus\bigcirc}$ formula φ over Π, there exists an NBA $\mathcal{A}_\varphi$ with alphabet $\Sigma = 2^\Pi$ that accepts all words and only those words over Π that satisfy φ, i.e., $\mathcal{L}_\omega(\mathcal{A}_\varphi) = Words(\varphi) = \{\sigma \in (2^\Pi)^\omega \mid \sigma \models \varphi\}$ [1], [19], [20]. Such $\mathcal{A}_\varphi$ can be automatically constructed using existing tools, such as LTL2BA [21], SPIN [22] and LBT [23], with the worst-case complexity that is exponential in the length of φ.

III. Problem Formulation

Consider a dynamical system $\mathbb{D}$ of the form (1) and a set $\Pi = \{p_0, p_1, \ldots, p_N\}$ of atomic propositions. For each atomic proposition p_i, we let $\mathcal{X}_i = [\![p_i]\!] \subseteq \mathcal{X}$ denote the set of states that satisfy p_i.

Problem statement: Given a specification φ expressed as an LTL$_{\setminus\bigcirc}$ formula over Π, determine if $\mathbb{D}$ satisfies φ.

Example 1: We use a simple problem to demonstrate the main ideas throughout the paper. Consider a two-dimensional system (which also appears in [24], [9]) governed by

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + \tfrac{1}{3}x_1(t)^3 - x_2(t), \end{aligned} \tag{6}$$

over the domain $\mathcal{X} = \{(x_1, x_2) \mid x_1^2 + x_2^2 \leq 49\}$ and let the regions of interest be given as

$$\begin{aligned} \mathcal{X}_0 &= \left\{(x_1, x_2) \mid (x_1+2)^2 + (x_2-4.5)^2 \leq 0.0625\right\}, \\ \mathcal{X}_1 &= \left\{(x_1, x_2) \mid (x_1-\sqrt{3})^2 + x_2^2 \leq 3\right\}, \\ \mathcal{X}_2 &= \left\{(x_1, x_2) \mid (x_1-4)^2 + (x_2-4)^2 \leq 1\right\}, \textit{and} \\ \mathcal{X}_3 &= \left\{(x_1, x_2) \mid x_1^2 + (x_2+3)^2 \leq 4\right\}. \end{aligned} \tag{7}$$

The phase portrait of (6) and the sets $\mathcal{X}$, $\mathcal{X}_0, \ldots, \mathcal{X}_3$ are shown in Figure 2. In this case, $\Pi = \{p_0, p_1, \ldots, p_3\}$, where for each $i \in \{0, \ldots, 3\}$, $[\![p_i]\!] = \mathcal{X}_i$.

We want to ensure that any trajectory of (6) satisfies the following conditions.

- Once it reaches $\mathcal{X}_2$, it cannot reach $\mathcal{X}_3$ forever.
- If it starts in $\mathcal{X}_0$, then it has to reach $\mathcal{X}_1$ before it reaches $\mathcal{X}_2$.

The property described above can be expressed as the LTL$_{\setminus\bigcirc}$ formula

$$\varphi = \Box(p_2 \implies \Box\neg p_3) \wedge \big(p_0 \implies (\Diamond p_2 \implies (\neg p_2 \,\mathcal{U}\, p_1))\big). \tag{8}$$

IV. Automata-Based Verification

Our approach to solve the LTL$_{\setminus\bigcirc}$ verification of dynamical systems defined in Section III relies on constructing a set $\Omega \subseteq (2^\Pi)^*$ of finite strings such that for any word $\sigma \in (2^\Pi)^\omega$, if $\sigma \not\models \varphi$, then there exists a substring $\omega \in \Omega$ of σ. Hence, to provide a proof of correctness of $\mathbb{D}$ with respect to φ, we "invalidate" each $\omega \in \Omega$ by showing that ω cannot be a substring of any word in $Trace(\mathbb{D})$.

To compute the set Ω, we first generate an NBA $\mathcal{A}_{\neg\varphi} = (Q, 2^\Pi, \delta, Q_0, F)$ that accepts all words and only those words over Π that satisfy $\neg\varphi$. It is well known from automata theory and model checking [1] that $Trace(\mathbb{D}) \nsubseteq Words(\varphi)$ if and only if there exists a word in $Trace(\mathbb{D})$ that is accepted by $\mathcal{A}_{\neg\varphi}$. Furthermore, there exists a word $\sigma \in (2^\Pi)^\omega$ that is accepted by $\mathcal{A}_{\neg\varphi}$ if and only if there exists a run of $\mathcal{A}_{\neg\varphi}$ of the form $q_0^p q_1^p \ldots q_{m_p}^p (q_0^c q_1^c \ldots q_{m_c}^c)^\omega$ where $m_p, m_c \in \mathbb{N}$ and $q_0^c \in F$.

Let $\mathcal{R}^{fin}$ be the set of finite run fragments of $\mathcal{A}_{\neg\varphi}$. In addition, for each $q, q' \in Q$, let $\mathcal{R}(q, q') \subseteq \mathcal{R}^{fin}$ be the set of finite run fragments of $\mathcal{A}_{\neg\varphi}$ that starts in q and ends in q'. Consider the set $\mathcal{R}^{acc} \subseteq \mathcal{R}^{fin}$ defined by $\mathcal{R}^{acc} = \{\pi^p \pi^c \mid \pi^p \in \mathcal{R}(q_0, q), \pi^c \in \mathcal{R}(q', q), q_0 \in Q_0, q \in F, q \xrightarrow{a} q' \text{ for some } a \in 2^\Pi\}$. Note that any run fragment in $\mathcal{R}^{acc}$ consists of two parts, π^p and π^c, where π^p corresponds to a finite run fragment from an initial state to an accepting state q of $\mathcal{A}_{\neg\varphi}$ and $q\pi^c$ corresponds to a finite run fragment from and to q, i.e., an accepting cycle starting with q. Finally, define Ω as the set of all finite strings generated by run fragments in $\mathcal{R}^{acc}$, i.e., $\Omega = \bigcup_{\pi \in \mathcal{R}^{acc}} \mathcal{ST}(\pi)$.

Example 2: Figure 3 shows an NBA $\mathcal{A}_{\neg\varphi}$ that accepts all and only words that satisfy $\neg\varphi$ where φ is defined in (8). Note that the transitions are simplified and only valid transitions, i.e., transitions (q, a, q') such that $[\![a]\!] \neq \emptyset$ are shown. From Figure 3, we get that $Q_0 = \{q_0\}$ and $F = \{q_4\}$. Hence, the set of run fragments from initial states to accepting states of $\mathcal{A}_{\neg\varphi}$ is given by $\mathcal{R}(q_0, q_4) = \{q_0 q_1^+ q_4^+, q_0 q_2^+ q_3^+ q_4^+, \; q_0 q_3^+ q_4^+\}$ and the set of accepting cycles of $\mathcal{A}_{\neg\varphi}$ is given by $\mathcal{R}(q_4, q_4) = \{q_4^+\}$. By appending run fragments in $\mathcal{R}(q_4, q_4)$ to those in $\mathcal{R}(q_0, q_4)$, we obtain $\mathcal{R}^{acc} = \{q_0 q_1^+ q_4 q_4^+, q_0 q_2^+ q_3^+ q_4 q_4^+, q_0 q_3^+ q_4 q_4^+\}$. Ω is then defined as the union of the following sets of finite strings:

- $\{a_{0,1} a_{1,1}^1 \ldots a_{1,1}^k a_{1,4} a_{4,4}^1 \ldots a_{4,4}^l \mid k \geq 0, l > 0, p_0 \in a_{0,1}, p_2 \in a_{1,4}, p_1 \notin a_{1,1}^j \text{ for all } j \in \{1, \ldots k\}\}$, which is generated by $q_0 q_1^+ q_4 q_4^+$,

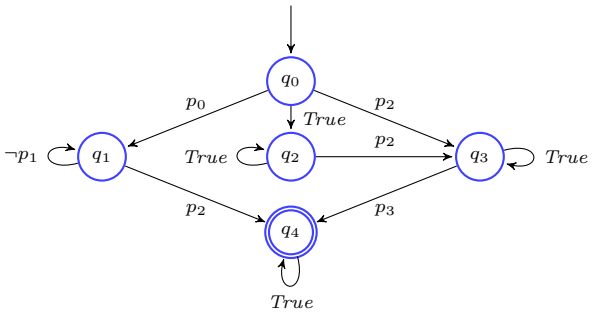

Fig. 3. NBA $\mathcal{A}_{\neg\varphi}$ that accepts all and only words that satisfy $\neg\varphi$ where φ is defined in (8). Note that the transitions are simplified and only valid transitions, i.e., transitions (q, a, q') such that $[\![a]\!] \neq \emptyset$ are shown. For example, the transition $(q_0, p_0 \wedge \neg p_1, q_1)$ is labeled with p_0 because $\mathcal{X}_0 \cap (\mathcal{X} \setminus \mathcal{X}_1) = \mathcal{X}_0$. An arrow without a source points to an initial state. An accepting state is drawn with a double circle.

- $\{a_{0,2}a^1_{2,2}\dots a^{k_1}_{2,2}a_{2,3}a^1_{3,3}\dots a^{k_2}_{3,3}a_{3,4}a^1_{4,4}\dots a^l_{4,4} \mid k_1, k_2 \geq 0, l > 0, p_2 \in a_{2,3}, p_3 \in a_{3,4}\}$, which is generated by $q_0q_2^+q_3^+q_4q_4^+$, and
- $\{a_{0,3}a^1_{3,3}\dots a^k_{3,3}a_{3,4}a^1_{4,4}\dots a^l_{4,4} \mid k \geq 0, l > 0, p_2 \in a_{0,3}, p_3 \in a_{3,4}\}$, which is generated by $q_0q_3^+q_4q_4^+$.

Lemma 2: For any infinite string $\sigma \in (2^\Pi)^\omega$, if $\sigma \not\models \varphi$, then there exists a substring $\omega \in \Omega$ of σ.

Proof: Consider an infinite string $\sigma = a_0a_1\dots \in (2^\Pi)^\omega$ such that $\sigma \not\models \varphi$. From automata theory [1], $\sigma \in \mathcal{L}_\omega(\mathcal{A}_{\neg\varphi})$; hence, there exists an accepting run $\pi = q_0q_1\dots$ for σ in $\mathcal{A}_{\neg\varphi}$. Since π is an accepting run, by definition, there exists $q \in F$ such that $q_i = q$ for infinitely many i. Let $j \geq 0$ and $k > j$ be indices such that $q_j = q_k = q$ and consider $\omega = a_0a_1\dots a_{k-1}$. Clearly, ω is a substring of σ. Furthermore, $q_0q_1\dots q_j \in \mathcal{R}(q_0, q)$ and $q_{j+1}q_{j+2}\dots q_k \in \mathcal{R}(q', q)$ where $q \xrightarrow{a_j} q'$. Thus, it is clear from the definition of $\mathcal{R}^{acc}$ that $\pi' = q_0q_1\dots q_k \in \mathcal{R}^{acc}$. Since $\omega \in \mathcal{ST}(\pi')$, we can conclude that $\omega \in \Omega$. ■

Example 3: Consider an infinite string $\sigma = a_0a_1\dots$ such that $p_2 \in a_i$ for some $i \in \mathbb{N}$ and $p_3 \in a_j$ for some $j > i$. It is obvious that $\sigma \not\models \Box(p_2 \implies \Box\neg p_3)$; hence, $\sigma \not\models \varphi$ where φ is defined in (8). Based on Lemma 2, there must exist a substring $\omega \in \Omega$ of σ. Consider a substring $\omega = a_{0,3}a^1_{3,3}\dots a^{j-i-1}_{3,3}a_{3,4}a_{4,4}$ of σ where $a_{0,3} = a_i$, $a^1_{3,3} = a_{i+1}, \dots, a^{j-i-1}_{3,3} = a_{j-1}$, $a_{3,4} = a_j$ and $a_{4,4} = a_{j+1}$. It is easy to check that $\omega \in \{a_{0,3}a^1_{3,3}\dots a^k_{3,3}a_{3,4}a^1_{4,4}\dots a^l_{4,4} \mid k \geq 0, l > 0, p_2 \in a_{0,3}, p_3 \in a_{3,4}\}$; hence from Example 2, $\omega \in \Omega$.

Lemma 3: Suppose for each $\omega \in \Omega$, there exists a substring ω' of ω such that ω' cannot be a substring of any word in $Trace(\mathbb{D})$. Then, $\mathbb{D}$ satisfies φ.

Proof: Assume, in order to establish a contradiction, that $\mathbb{D}$ does not satisfy φ. Then, there exists a trajectory x of $\mathbb{D}$ and its trace σ_x such that $\sigma_x \not\models \varphi$. From Lemma 2, there exists a substring $\omega \in \Omega$ of σ_x. However, since $\omega \in \Omega$, there exists a substring ω' of ω that is not a substring of σ_x. Hence, ω cannot be a substring of σ_x, leading to a contradiction. ■

Based on Lemma 3, we can verify that $\mathbb{D}$ satisfies φ by checking that for each $\omega \in \Omega$, there exists a substring of ω that cannot be a substring of any word in $Trace(\mathbb{D})$. However, since $\mathcal{R}^{acc}$ is, in general, not finite, Ω is also, in general, not finite (as illustrated in Example 2). As a result, invalidating all $\omega \in \Omega$ may not be straightforward. In the next section, we propose a finite collection $\Pi_1, \Pi_2, \dots, \Pi_M$ of representative sets of finite run fragments with the property that for each $\omega \in \Omega$, there exists some $i \in \{1, \dots, M\}$, such that each $\pi \in \Pi_i$ can be used to "derive" a substring of ω that is in a certain form. (We will make it clear later how such a substring can be derived.) Hence, invalidating all strings derived from some $\pi \in \Pi_i$ for each $i \in \{1, \dots, M\}$ provides a certificate of system correctness with respect to φ. Then, in Section VI, we show that due to their particular form, the strings derived from any $\pi \in \Pi_i$, $i \in \{1, \dots, M\}$ are amenable to verification based on the idea of barrier certificates and to algorithmic solutions, for the cases where the vector field in (1) and the sets $\mathcal{X}, \mathcal{X}_0, \dots, \mathcal{X}_N$ can be described by polynomial or rational functions, through sum-of-squares relaxations for polynomial optimization.

To recap, based on the definition of a trace, the behavior of $\mathbb{D}$ is formalized by the sequences of subsets of $\mathcal{X}$ visited along its trajectories. These subsets of $\mathcal{X}$ are constructed from a collection of sets $\mathcal{X}_1, \dots, \mathcal{X}_N$; hence, each of them captures certain characteristics of $\mathbb{D}$ over $\mathcal{X}$ as described by a boolean combination of atomic propositions in Π. The language $\mathcal{L}_\omega(\mathcal{A}_{\neg\varphi})$ accepted by $\mathcal{A}_{\neg\varphi}$ essentially describes the sequences of subsets of $\mathcal{X}$ that violate φ. Hence, to prove that $\mathbb{D}$ satisfies φ, we show that for each of its trajectories and for each sequence in $\mathcal{L}_\omega(\mathcal{A}_{\neg\varphi})$, there exists a portion of the sequence that the trajectory cannot follow.

V. Representative Sets of Run Fragments

Let $\mathcal{G} = (V^\mathcal{G}, E^\mathcal{G})$ denote the underlying directed graph of $\mathcal{A}_{\neg\varphi}$, i.e., $V^\mathcal{G} = Q$ and $E^\mathcal{G} \subseteq V^\mathcal{G} \times V^\mathcal{G}$ such that $(q, q') \in E^\mathcal{G}$ if and only if there exists $a \in 2^\Pi$ such that $q \xrightarrow{a} q'$. A path in $\mathcal{G}$ is a finite or infinite sequence π of states such that for any two consecutive states q, q' in π, $(q, q') \in E^\mathcal{G}$. From the construction of $\mathcal{G}$, it is obvious that π is a path in $\mathcal{G}$ if and only if it is a run fragment of $\mathcal{A}_{\neg\varphi}$. Given a finite path $\pi = q_0q_1\dots q_m$ or an infinite path $\pi = q_0q_1\dots$, a subpath of π is any finite path of the form $q_iq_{i+1}\dots q_{i+k}$ where $i, k \geq 0$ and $i + k \leq m$ if π is finite.

A variant of depth-first search [25] provided in Algorithm V can be used to find the set of all the paths from a state q to a state q' with no repeated edges and no consecutive repetitions of states in $\mathcal{G}$, including the case where $q = q'$. Since $E^\mathcal{G}$ is finite, the set of all the paths from q to q' with no repeated edges and no consecutive repetitions of states is finite for any $q, q' \in Q$ (unlike the set of all the paths from q to q' which may not be finite as these paths may contain cycles that can be repeated arbitrary times). As will be discussed later, such a set of paths with no repeated edges and no consecutive repetitions of states can be used to form a finite set $\mathcal{SP}$ of subpaths from q to q', each of which can be "extended" to a subpath of any path from q to q'. Proposition 1, presented later, provides an exact definition of "extending" a path.

Algorithm 1 $\texttt{DFS}(\mathcal{G}, q, q')$

$\mathcal{P}^{\mathcal{G}}_{q,q'} \leftarrow \emptyset$
$toVisit \leftarrow \{q\}$
$paths \leftarrow \{q\}$
if $q' = q$ **then**
  Append q to $\mathcal{P}^{\mathcal{G}}_{q,q'}$
end if
while $toVisit \neq \emptyset$ **do**
  Remove the last element of $toVisit$ and assign it to v
  Remove the last sequence in $paths$ and assign it to $path2v$
  for all $nb \neq v$ such that $(v, nb) \in E^{\mathcal{G}}$ **do**
    if $nb = q'$ **then**
      Append the sequence obtained by concatenating $path2v$ and nb to $\mathcal{P}^{\mathcal{G}}_{q,q'}$
    else if v is not followed by *nb* in *path2v* **then**
      Append *nb* to *toVisit*
      Append the sequence obtained by concatenating *path2v* and nb to *paths*;
    end if
  end for
end while
return $\mathcal{P}^{\mathcal{G}}_{q,q'}$

Given $q, q' \in Q$, let $\mathcal{P}(q, q')$ be the set of paths from a state q to a state q' with no repeated edges and no consecutive repetitions of states in $\mathcal{G}$. In addition, for each $q \in F$, let $\mathcal{P}^{path}(q) = \{\pi \in \mathcal{P}(q_0, q) \mid q_0 \in Q_0\}$ be the set of paths from an initial state of $\mathcal{A}_{\neg\varphi}$ to q with no repeated edges and no consecutive repetitions of states and let $\mathcal{P}^{cyc}(q) = \{\pi \in \mathcal{P}(q, q) \mid \mathcal{P}^{path}(q) \neq \emptyset$ and if $\pi = q$, then $(q, q) \in E^{\mathcal{G}}\}$ be the set of reachable cycles that start from q and have no repeated edges or consecutive repetitions of states. From the definition of $\mathcal{P}(\cdot,\cdot)$ and $\mathcal{R}(\cdot,\cdot)$, it is obvious that for each $q \in F$, $\mathcal{P}^{cyc}(q)$ and $\mathcal{P}^{path}(q)$ are finite, $\mathcal{P}^{cyc}(q) \subseteq \mathcal{R}(q, q)$ and $\mathcal{P}^{path}(q) \subseteq \bigcup_{q_0 \in Q_0} \mathcal{R}(q_0, q)$. In this section, we show that a collection $\Pi_1, \Pi_2, \ldots, \Pi_M$ of representative sets of finite run fragments as described at the end of Section IV can be constructed from $\mathcal{P}^{cyc}(q)$ and $\mathcal{P}^{path}(q)$ for each $q \in F$.

For a finite path π in $\mathcal{G}$, we define $\mathcal{PF}^3(\pi)$ as the set of all subpaths of π with length 3, i.e., $\mathcal{PF}^3(q_0 q_1 \ldots q_m) = \{q_i q_{i+1} q_{i+2} \mid 0 \leq i \leq m-2\}$. Note that for a path π with length less than 3, $\mathcal{PF}^3(\pi) = \emptyset$.

Example 4: Let $\mathcal{A}_{\neg\varphi}$ be the NBA shown in Figure 3. Then, $F = \{q_4\}$. Applying Algorithm V, we get

$$\begin{aligned}
\mathcal{P}^{cyc}(q_4) &= \{q_4\},\\
\mathcal{P}^{path}(q_4) &= \{q_0q_1q_4, q_0q_2q_3q_4, q_0q_3q_4\},\\
\mathcal{PF}^3(q_4) &= \emptyset,\\
\mathcal{PF}^3(q_0q_1q_4) &= \{q_0q_1q_4\},\\
\mathcal{PF}^3(q_0q_2q_3q_4) &= \{q_0q_2q_3, q_2q_3q_4\},\\
\mathcal{PF}^3(q_0q_3q_4) &= \{q_0q_3q_4\}.
\end{aligned}$$

Note that any $\omega \in \Omega$ can be written as $\omega = \omega^p\omega^c$ where ω^p and ω^c are generated from π^p and $q\pi^c$, respectively, for some $\pi^p\pi^c \in \mathcal{R}^{acc}$ where π^p corresponds to a finite run fragment from an initial state to an accepting state q of $\mathcal{A}_{\neg\varphi}$ and $q\pi^c$ corresponds to an accepting cycle of $\mathcal{A}_{\neg\varphi}$. Hence, to invalidate ω, we can invalidate either ω^p or ω^c. As will be shown in Proposition 1, for any path π from q to q', there exists $\pi' \in \mathcal{P}(q, q')$ such that $\mathcal{SP} = \mathcal{PF}^3(\pi')$ is a finite set of paths, each of which can be extended to a subpath of π. Hence, a way to invalidate ω^c is to show that for each $p \in \mathcal{P}^{cyc}(q)$, there exists $\tilde{\pi} \in \mathcal{PF}^3(p)$ such that all finite strings generated by each extension of $\tilde{\pi}$ cannot be a substring of any word of $\mathbb{D}$. Similarly, a way to invalidate ω^p is to show that for each $p \in \mathcal{P}^{path}(q)$, there exists $\tilde{\pi} \in \mathcal{PF}^3(p)$ such that all finite strings generated by each extension of $\tilde{\pi}$ cannot be a substring of any word of $\mathbb{D}$.

Proposition 1: Suppose for each $q \in F$, either of the following conditions (1) and (2) holds:

(1) For each $p \in \mathcal{P}^{cyc}(q)$, there exists $\pi = q_0q_1q_2 \in \mathcal{PF}^3(p)$ such that
  (a) all finite strings $a_0a_1 \in \mathcal{ST}(\pi)$ cannot be a substring of any word in $Trace(\mathbb{D})$, and
  (b) if $(q_1, q_1) \in E^{\mathcal{G}}$, then all finite strings $a_0\tilde{a}_0 \ldots \tilde{a}_k a_1 \in \mathcal{ST}(q_0q_1q_1^+q_2)$, $k \in \mathbb{N}$ cannot be a substring of any word in $Trace(\mathbb{D})$.

(2) For each $p \in \mathcal{P}^{path}(q)$, there exists $\pi = q_0q_1q_2 \in \mathcal{PF}^3(p)$ such that
  (a) all finite strings $a_0a_1 \in \mathcal{ST}(\pi)$ cannot be a substring of any word in $Trace(\mathbb{D})$, and
  (b) if $(q_1, q_1) \in E^{\mathcal{G}}$, then all finite strings $a_0\tilde{a}_0 \ldots \tilde{a}_k a_1 \in \mathcal{ST}(q_0q_1q_1^+q_2)$, $k \in \mathbb{N}$ cannot be a substring of any word in $Trace(\mathbb{D})$.

Then, $\mathbb{D}$ satisfies φ.

Proof: Consider an arbitrary finite string $\omega \in \Omega$. From the definition of Ω, there exist an accepting state $q \in F$ and a finite run fragment of the form $q_0^pq_1^p \ldots q_{m_p}^pqq_0^cq_1^c \ldots q_{m_c}^cq$ where $m_p, m_c \in \mathbb{N}$ and $q_0^p \in Q_0$ from which ω is generated. Let $\pi^p = q_0^pq_1^p \ldots q_{m_p}^pq$ and $\pi^c = qq_0^cq_1^c \ldots q_{m_c}^cq$. In addition, let ω^p and ω^c be the substrings of ω that are generated from π^p and π^c, respectively. Note that both π^p and π^c correspond to paths in $\mathcal{G}$. To prove that satisfying either condition (1) or (2) ensures the correctness of $\mathbb{D}$ with respect to φ, we show that both of the following conditions hold.

(i) There exists $p \in \mathcal{P}^{cyc}(q)$ such that for each $\pi = q_0q_1q_2 \in \mathcal{PF}^3(p)$, if $(q_1, q_1) \notin E^{\mathcal{G}}$, then π^c contains π; otherwise π^c contains some run fragment of the form $q_0q_1^+q_2$.

(ii) There exists $p \in \mathcal{P}^{path}(q)$ such that for each $\pi = q_0q_1q_2 \in \mathcal{PF}^3(p)$, if $(q_1, q_1) \notin E^{\mathcal{G}}$, then π^p contains π; otherwise π^p contains some run fragment of the form $q_0q_1^+q_2$.

Thus, satisfying condition (1) ensures that there exists a substring $\omega^{c'}$ of ω^c such that $\omega^{c'}$ cannot be a substring of any word in $Trace(\mathbb{D})$. Since ω^c is a substring of ω, $\omega^{c'}$ is also a substring of ω. We can then conclude from Lemma 3 that $\mathbb{D}$ satisfies φ. Similarly, satisfying condition (2) ensures that

there exists a substring $\omega^{p'}$ of ω^p, which is also a substring of ω, that cannot be a substring of any word in $Trace(\mathbb{D})$. Lemma 3 can then be applied to conclude that $\mathbb{D}$ satisfies φ.

First, consider condition (i) and the case where π^p does not contain any repeated edges or consecutive repetitions of states in $\mathcal{G}$. In this case, it directly follows from the definition of $\mathcal{P}^{path}$ that $\pi^p \in \mathcal{P}^{path}(q)$; hence, condition (i) is trivially satisfied. Next, consider the case where π^p contains a repeated edge, i.e., there exist $\tilde{q}_1, \tilde{q}_2 \in Q$ such that $\tilde{q}_1$ is followed by $\tilde{q}_2$ more than once in π^p. Then, π^p must contain a subsequence of the form $\tilde{q}_1\tilde{q}_2 \dots \tilde{q}_1\tilde{q}_2$. Let $\pi^{p'}$ be a run fragment that is obtained from π^p by replacing this subsequence with $\tilde{q}_1\tilde{q}_2$; thus, removing a repeated edge $(\tilde{q}_1, \tilde{q}_2)$ from π^p. It can be checked that for any $\pi = q_0q_1q_2 \in \mathcal{PF}^3(\pi^{p'})$, if $(q_1, q_1) \notin E^{\mathcal{G}}$, then $\pi \in \mathcal{PF}^3(\pi^p)$; otherwise, π^p contains a subsequence of the form $q_0q_1^+q_2$. For the case where π^p contains a consecutive repetition of some state $\tilde{q} \in Q$, i.e., $\pi^p = q_0^p q_1^p \dots \tilde{q}\tilde{q} \dots \tilde{q} \dots q_{m_p}^p q$, we construct $\pi^{p''} = q_0^p q_1^p \dots \tilde{q} \dots q_{m_p}^p q$ by removing such a consecutive repetition of $\tilde{q}$. It can be easily checked that for any $\pi = q_0q_1q_2 \in \mathcal{PF}^3(\pi^{p''})$, if $(q_1, q_1) \notin E^{\mathcal{G}}$, then $\pi \in \mathcal{PF}^3(\pi^p)$; otherwise, π^p contains a subsequence of the form $q_0q_1^+q_2$. We apply this process of removing repeated edges and consecutive repetitions of states in π^p until we obtain a run fragment $\tilde{\pi}^p$ that does not contain any repeated edges or consecutive repetitions of states. Then, $\tilde{\pi}^p \in \mathcal{P}^{path}(q)$ and for any $\pi = q_0q_1q_2 \in \mathcal{PF}^3(\tilde{\pi}^p)$, if $(q_1, q_1) \notin E^{\mathcal{G}}$, then $\pi \in \mathcal{PF}^3(\pi^p)$; otherwise, π^p contains a subsequence a of the form $q_0q_1^+q_2$. Condition (ii) can be treated in a similar way. ∎

To sum, Proposition 1 provides a sufficient (but not necessary) condition for verifying that no word in $Trace(\mathbb{D})$ is accepted by $\mathcal{A}_{\neg\varphi}$. Based on Proposition 1, we construct sets $\mathcal{PF}_1^{cyc,q}, \mathcal{PF}_2^{cyc,q}, \dots, \mathcal{PF}_{M_c}^{cyc,q}$ and $\mathcal{PF}_1^{path,q}, \mathcal{PF}_2^{path,q}, \dots, \mathcal{PF}_{M_p}^{path,q}$ for each $q \in F$ where M_c is the cardinality of $\mathcal{P}^{cyc}(q)$, M_p is the cardinality of $\mathcal{P}^{path}(q)$, for each $i \in \{1, \dots, M_c\}$, $\mathcal{PF}_i^{cyc,q} = \mathcal{PF}^3(p)$, p is the ith path in $\mathcal{P}^{cyc}(q)$ and for each $i \in \{1, \dots, M_p\}$, $\mathcal{PF}_i^{path,q} = \mathcal{PF}^3(p)$, p is the ith path in $\mathcal{P}^{path}(q)$. Then, we show that for each $q \in F$, either (1) for each $i \in \{1, \dots, M_c\}$, there exists $\pi \in \mathcal{PF}_i^{cyc,q}$ such that all finite strings generated by each extension of π as described in conditions (1)-(a) and (1)-(b) cannot be a substring of any word in $Trace(\mathbb{D})$, hence, invalidating all accepting cycles starting with q, or (2) for each $i \in \{1, \dots, M_p\}$, there exists $\pi \in \mathcal{PF}_i^{path,q}$ such that all finite strings generated by each extension of π as described in conditions (2)-(a) and (2)-(b) cannot be a substring of any word in $Trace(\mathbb{D})$, hence, invalidating all paths to the accepting state q.

In the next section, we discuss a set of conditions whose satisfaction implies the satisfaction of the conditions in (1) and (2) of Proposition 1. The satisfaction of these new conditions can be verified algorithmically; hence, their verification is amenable to automation.

VI. Barrier Certificates for Invalidating Substrings

Conditions (1) and (2) of Proposition 1 require considering finite strings of the form a_0a_1 and $a_0\tilde{a}_0 \dots \tilde{a}_k a_1$ where $k \in \mathbb{N}$ and $a_0, a_1, \tilde{a}_0, \dots, \tilde{a}_k \in 2^{\Pi}$. Lemma 4 and Lemma 5 provide a necessary condition for a trajectory of $\mathbb{D}$ to have a trace with a substring of the form a_0a_1 and $a_0\tilde{a}_0 \dots \tilde{a}_k a_1$, $k \in \mathbb{N}$, respectively.

Lemma 4: Consider $\Sigma_0, \Sigma_1 \subseteq 2^{\Pi}$ and a set $\tilde{\Omega} = \{a_0a_1 \mid a_0 \in \Sigma_0, a_1 \in \Sigma_1\}$ of finite strings. Let $\mathcal{Y}_0 = \bigcup_{a\in\Sigma_0}[\![a]\!]$ and $\mathcal{Y}_1 = \bigcup_{a\in\Sigma_1}[\![a]\!]$. If there exists a trajectory x of $\mathbb{D}$ such that some finite string in $\tilde{\Omega}$ is a substring of a trace of x, then there exist $t_1 > t_0 \geq 0$ and $t_0' \in [t_0, t_1]$ such that $x(t) \in \mathcal{Y}_0$ for all $t \in [t_0, t_0')$, $x(t) \in \mathcal{Y}_1$ for all $t \in (t_0', t_1]$ and $x(t_0') \in \mathcal{Y}_0 \cup \mathcal{Y}_1$.

Proof: This follows directly from the definition of trace. ∎

Lemma 5: Consider $\Sigma_0, \Sigma_1, \tilde{\Sigma} \subseteq 2^{\Pi}$ and a set $\tilde{\Omega} = \{a_0\tilde{a}_0 \dots \tilde{a}_k a_1 \mid k \in \mathbb{N}, a_0 \in \Sigma_0, \tilde{a}_0, \dots, \tilde{a}_k \in \tilde{\Sigma}, a_1 \in \Sigma_1\}$ of finite strings. Let $\mathcal{Y}_0 = \bigcup_{a\in\Sigma_0}[\![a]\!]$, $\mathcal{Y}_1 = \bigcup_{a\in\Sigma_1}[\![a]\!]$ and $\tilde{\mathcal{Y}} = \bigcup_{a\in\tilde{\Sigma}}[\![a]\!]$. If there exists a trajectory x of $\mathbb{D}$ such that some finite string in $\tilde{\Omega}$ is a substring of a trace of x, then there exists $t_1 > t_0 \geq 0$ such that $x(t_0) \in \mathcal{Y}_0$, $x(t_1) \in \mathcal{Y}_1$ and $x(t) \in \mathcal{Y}$ for all $t \in [t_0, t_1]$ where $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1 \cup \tilde{Y}$.

Proof: Consider a trajectory x of $\mathbb{D}$ and a finite substring $\sigma = a_0\tilde{a}_0 \dots \tilde{a}_k a_1$ where $k \in \mathbb{N}$ and $a_0 \in \Sigma_0$, $\tilde{a}_0, \dots, \tilde{a}_k \in \tilde{\Sigma}$ and $a_1 \in \Sigma_1$. Suppose σ is a substring of a trace of x. Then, from the definition of trace, we can conclude that there exist $t_1 > t_0 \geq 0$ such that $x(t_0) \in [\![a_0]\!]$, $x(t_1) \in [\![a_1]\!]$ and for all $t \in [t_0, t_1]$, $x(t) \in [\![a_0]\!] \cup [\![\tilde{a}_0]\!] \cup \dots \cup [\![\tilde{a}_k]\!] \cup [\![a_1]\!]$, i.e., $x(t_0) \in \mathcal{Y}_0$, $x(t_1) \in \mathcal{Y}_1$ and $x(t) \in \mathcal{Y}$ for all $t \in [t_0, t_1]$. ∎

We now consider conditions (1)-(a) and (2)-(a) of Proposition 1, which require considering a finite string of the form a_0a_1 where $a_0, a_1 \in 2^{\Pi}$. The following lemma provides a sufficient condition, based on checking the emptiness of set intersection, for validating that such a finite string cannot be a substring of any word in $Trace(\mathbb{D})$.

Lemma 6: Consider $\Sigma_0, \Sigma_1 \subseteq 2^{\Pi}$ and a set $\tilde{\Omega} = \{a_0a_1 \mid a_0 \in \Sigma_0, a_1 \in \Sigma_1\}$ of finite strings. Let $\mathcal{Y}_0 = \bigcup_{a\in\Sigma_0}[\![a]\!]$ and $\mathcal{Y}_1 = \bigcup_{a\in\Sigma_1}[\![a]\!]$. Suppose $\overline{\mathcal{Y}_0} \cap \overline{\mathcal{Y}_1} = \emptyset$. Then, no finite string in $\tilde{\Omega}$ can be a substring of any word in $Trace(\mathbb{D})$.

Proof: Suppose, in order to establish a contradiction, that there exists a trajectory x of $\mathbb{D}$ such that some $\omega = a_0a_1 \in \tilde{\Omega}$ is a substring of a trace of x. From Lemma 4, there must exist $t_1 > t_0 \geq 0$ and $t_0' \in [t_0, t_1]$ such that $x(t) \in \mathcal{Y}_0$ for all $t \in [t_0, t_0')$ and $x(t) \in \mathcal{Y}_1$ for all $t \in (t_0', t_1]$. Furthermore, from the continuity of the trajectories of (1), $x(t) \in \mathcal{Y}_0$ for all $t \in [t_0, t_0')$ implies that $x(t) \in \overline{\mathcal{Y}_0}$ for all $t \in [t_0, t_0']$. Similarly, $x(t) \in \mathcal{Y}_1$ for all $t \in (t_0', t_1]$ implies that $x(t) \in \overline{\mathcal{Y}_1}$ for all $t \in [t_0', t_1]$. As a result, it must be the case that $x(t_0') \in \overline{\mathcal{Y}_0}$ and $x(t_0') \in \overline{\mathcal{Y}_1}$, and hence $x(t_0') \in \overline{\mathcal{Y}_0} \cap \overline{\mathcal{Y}_1}$, leading to a contradiction. ∎

Using the notion of barrier certificate [26], [27], [9], we provide a sufficient condition for checking that conditions

(1) and (2) of Proposition 1 are satisfied. First, Corollary 1 combines Lemma 1 and Lemma 4 to provide a sufficient condition for validating that a finite string of the form a_0a_1 where $a_0, a_1 \in 2^{\Pi}$ cannot be a substring of any word in $Trace(\mathbb{D})$.

Corollary 1: Consider $\Sigma_0, \Sigma_1 \subseteq 2^{\Pi}$ and a set $\tilde{\Omega} = \{a_0a_1 \mid a_0 \in \Sigma_0, a_1 \in \Sigma_1\}$ of finite strings. Let $\mathcal{Y}_0 = \bigcup_{a\in\Sigma_0}[\![a]\!]$, $\mathcal{Y}_1 = \bigcup_{a\in\Sigma_1}[\![a]\!]$ and $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1$. Suppose there exists a differentiable function $B \mid \mathcal{X} \to \mathbb{R}$ satisfying conditions (2)-(4). Then, no finite string in $\tilde{\Omega}$ can be a substring of any word in $Trace(\mathbb{D})$.

Finally, the following corollary combines Lemma 1 and Lemma 5 to provide a sufficient condition for validating that a finite string of the form $a_0\tilde{a}_0 \dots \tilde{a}_k a_1$ where $k \in \mathbb{N}$ and $a_0, a_1, \tilde{a}_0, \dots, \tilde{a}_k \in 2^{\Pi}$ cannot be a substring of any word in $Trace(\mathbb{D})$.

Corollary 2: Consider $\Sigma_0, \Sigma_1, \tilde{\Sigma} \subseteq 2^{\Pi}$ and a set $\tilde{\Omega} = \{a_0\tilde{a}_0 \dots \tilde{a}_k a_1 \mid k \in \mathbb{N}, a_0 \in \Sigma_0, \tilde{a}_0, \dots, \tilde{a}_k \in \tilde{\Sigma}, a_1 \in \Sigma_1\}$ of finite strings. Let $\mathcal{Y}_0 = \bigcup_{a\in\Sigma_0}[\![a]\!]$ $\mathcal{Y}_1 = \bigcup_{a\in\Sigma_1}[\![a]\!]$, $\tilde{\mathcal{Y}} = \bigcup_{a\in\tilde{\Sigma}}[\![a]\!]$ and $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1 \cup \tilde{\mathcal{Y}}$. Suppose there exists a differentiable function $B : \mathcal{X} \to \mathbb{R}$ satisfying conditions (2)-(4). Then, no finite string in $\tilde{\Omega}$ can be a substring of any word in $Trace(\mathbb{D})$.

VII. LTL$_{\setminus\bigcirc}$ Verification Procedure

Based on the results presented in Section V and Section VI, we propose the following procedure for LTL$_{\setminus\bigcirc}$ verification of dynamical systems.

1) Compute $\mathcal{A}_{\neg\varphi}$.
2) Compute $\mathcal{P}^{cyc}(q)$ and $\mathcal{P}^{path}(q)$ for each $q \in F$ using Algorithm V.
3) For each $q \in F$, carry out the following steps.
   a) Generate $\mathcal{PF}^3(c)$ for each $c \in \mathcal{P}^{cyc}(q)$ and $\mathcal{PF}^3(p)$ for each $p \in \mathcal{P}^{path}(q)$. (From its definition, $\mathcal{PF}^3(\pi)$ can be easily generated for any given finite path π in $\mathcal{G}$.)
   b) Check whether condition (1) or condition (2) of Proposition 1 is satisfied. Conditions (1)-(a) and (2)-(a) can be checked using Lemma 6 or Corollary 1 whereas conditions (1)-(b) and (2)-(b) can be checked using using Corollary 2.
      - If either condition (1) or condition (2) holds, continue to process next accepting state $q \in F$ or terminate and report that $\mathbb{D}$ satisfies φ if all $q \in F$ has been processed.
      - Otherwise, terminate and report the failure for determining whether $\mathbb{D}$ satisfies φ using this procedure.

Steps 1-3(a) above can be automated. For example, off-the-shelf tools such as LTL2BA, SPIN and LBT can be used to compute of $\mathcal{A}_{\neg\varphi}$ in step 1. Checking conditions (1)-(a) and (2)-(a) of Proposition 1 can be automated based on Lemma 6 by employing generalizations of the so-called S-procedure [11] or special cases of the Positivstellensatz [10], [28]. Furthermore, if the sets $\mathcal{X}, \mathcal{X}_0, \dots, \mathcal{X}_N$ can be described by polynomial functions, then verification of the conditions in Corollary 1 and Corollary 2 can be reformulated (potentialy conservatively) as sum-of-squares feasibility problems [10], [29]. Specifically, Lemma 7 provides a set of sufficient conditions for the existence of a barrier certificate B as required by Lemma 1 to determine whether condition (1) or condition (2) of Proposition 1 is satisfied.

Lemma 7: Let $\mathcal{Y}, \mathcal{Y}_0, \mathcal{Y}_1 \subseteq \mathcal{X}$. Assume that $\overline{\mathcal{Y}_0}$ and $\overline{\mathcal{Y}_1}$ can be defined by the inequality $g_0(x) \geq 0$ and $g_1(x) \geq 0$, respectively, i.e., $\overline{\mathcal{Y}_0} = \{x : \mathbb{R}^n \mid g_0(x) \geq 0\}$ and $\overline{\mathcal{Y}_1} = \{x : \mathbb{R}^n \mid g_1(x) \geq 0\}$. Additionally, assume that $\overline{\mathcal{Y}}$ can be defined by the inequality $g(x) \geq 0$. Suppose there exist a polynomial B, a constant $\epsilon > 0$ and sum-of-squares polynomials s_0, s_1, s_2 and s_3 such that the following expressions are sum-of-squares polynomials

$$-B(x) - s_0(x)g_0(x), \tag{9}$$

$$B(x) - \epsilon - s_1(x)g_1(x), and \tag{10}$$

$$-\frac{\partial B}{\partial x}(x)f(x) - s_2(x)g(x) + s_3(x)g_1(x). \tag{11}$$

Then, B satisfies conditions (2)-(4).

Proof: Consider an arbitrary $x \in \mathcal{Y}_0$. Then, $g_0(x) \geq 0$. Furthermore, since (9) and $s_0(x)$ are sum-of-squares polynomials, we get that $-B(x) - s_0(x)g_0(x) \geq 0$ and $s_0(x) \geq 0$. Combining this with $g_0(x) \geq 0$, we obtain $B(x) \leq 0$, satisfying (2). Similarly, we can show that (11) being a sum-of-squares polynomial ensures that (4) is satisfied. Finally, consider (10) and an arbitrary $x \in \overline{\mathcal{Y}_1}$. Using the same argument as before, we get $B(x) - \epsilon \geq 0$. Since $\epsilon > 0$, we obtain $B(x) > 0$, satisfying (3). ∎

Based on Lemma 7, a function $B : \mathcal{X} \to \mathbb{R}$ satisfying conditions (2)-(4) can be automatically computed by solving the sum-of-squares problem in Lemma 7, which is convex and can be parsed, using SOSTOOLS [30] and SOSOPT [31], into a semidefinite program, provided that the vector field f is polynomial or rational. Note that in Lemma 7, we assume that $\overline{\mathcal{Y}}$, $\overline{\mathcal{Y}_0}$ and $\overline{\mathcal{Y}_1}$ can be described by polynomial functions g, g_0 and g_1, respectively, for the ease of the presentation. The result, however, can be easily extended to handle the case where each of these sets are described by a set of polynomial functions. For example, suppose $\overline{\mathcal{Y}_0} = \{x : \mathbb{R}^n \mid g_{0,1}(x) \geq 0, \dots, g_{0,k}(x) \geq 0\}$ where $k \in \mathbb{N}$ and $g_{0,1}, \dots, g_{0,k}$ are polynomial functions. Then, we need to find sum-of-squares polynomials $s_{0,1}, \dots, s_{0,k}$, rather than only s_0. In addition, rather than requiring that (9) is a sum-of-squares polynomial, we require that $-B(x) - s_{0,1}(x)g_{0,1}(x) - \dots - s_{0,k}(x)g_{0,k}(x)$ is a sum-of-squares polynomial. The case where other sets are described by a set of polynomial functions can be treated in a similar way.

VIII. Discussion

A. Sources of Incompleteness

The LTL$_{\setminus\bigcirc}$ verification procedure developed in the previous sections is sound but not complete, i.e., if it reports that $\mathbb{D}$ satisfies φ, then we can correctly conclude that $\mathbb{D}$ actually satisfies φ. However, if it reports failure, then $\mathbb{D}$ may or may not satisfy φ. The incompleteness is due to various

sources of conservatism included in the procedure for LTL$_{\setminus\bigcirc}$ verification of dynamical systems proposed in Section VII.

First, Proposition 1 provides only a sufficient condition for verifying that for each $\omega \in \Omega$, where Ω is as defined in Section IV, there exists a substring ω' of ω that cannot be a substring of any word in $Trace(\mathbb{D})$. However, such a sufficient condition only considers substrings ω' that are in a particular form since it may not be possible to check all the substrings of all $\omega \in \Omega$ due to the possible infiniteness of Ω. We provide further discussion on this issue in Section VIII-D. Another source of conservatism comes from Lemma 6, Corollary 1 and Corollary 2, which only provide sufficient conditions for verifying that no finite string in the particular form considered in Proposition 1 can be a substring of any word in $Trace(\mathbb{D})$. Finally, Lemma 7 introduces another source of conservatism as only a sufficient condition for the existence of a function $B : \mathcal{X} \to \mathbb{R}$ satisfying conditions (2)-(4) is provided. The conservatism due to this final cause may be reduced by searching for polynomial barrier certificates (B) and S-procedure multipliers (s_0, s_1, s_2 and s_3) of higher degrees.

B. Computational Complexity

Let $\mathcal{A}_{\neg\varphi} = (Q, 2^{\Pi}, \delta, Q_0, F)$. It can be shown [1] that the size $|Q|$ is at most $|\neg\varphi|2^{|\neg\varphi|}$ where $|\neg\varphi|$ is the length (in terms of the number of operations) of $\neg\varphi$. (In practice, the size $|Q|$ is typically much smaller than this upper limit [32].) Let $|E^{\mathcal{G}}|$ represent the number of edges of $\mathcal{G}$. Note that from the construction of $\mathcal{G}$ as explained in Section V, $|E^{\mathcal{G}}| \leq |Q|^2$ and $|E^{\mathcal{G}}| \leq |\delta|$ where $|\delta|$ is the number of transitions in $\mathcal{A}_{\neg\varphi}$. In the worst case, for each $q \in F$, the size of $\mathcal{P}^{cyc}(q)$ is $(|Q|-1)^{|E^{\mathcal{G}}|-1}$ whereas the size of $\mathcal{P}^{path}(q)$ is $|Q_0|(|Q|-1)^{|E^{\mathcal{G}}|-1}$. (Roughly, this is because the length of each path in $\mathcal{P}^{cyc}(q)$ and $\mathcal{P}^{path}(q)$ is at most $|E^{\mathcal{G}}|+1$ since edges cannot be repeated. In addition, at each state except the last two states in the path, there are $|Q|-1$ possibilities of the next state since consecutive repetitions of states are not allowed.) As a result, for each $q \in F$, the total of at most $(|E^{\mathcal{G}}|-1)(|Q|-1)^{|E^{\mathcal{G}}|-1}(1+|Q_0|)$ subpaths of length 3 need to be considered in Step (3)-(b) of the LTL$_{\setminus\bigcirc}$ verification procedure described in Section VII. Note that each of these subpaths corresponds to a numerical search for a barrier certificate and S-procedure multipliers that satisfy the conditions in Lemma 7. For the largest degree of the polynomials in (9)-(11) and the number n of continuous states, the complexity of this search is polynomial in each when the other fixed.

C. Comparison to Approaches Based on Explicit Discretization of Dynamics

A common approach for verifying dynamical systems (call $\mathbb{D}$) subject to LTL$_{\setminus\bigcirc}$ specifications (call φ) is to explicitly construct a finite state abstraction $\mathbb{T}$ of $\mathbb{D}$ [3], [4]. We now briefly compare our method to such approaches with respect to their (in)completeness, computational cost, and conservatism.

Except for certain special cases, $\mathbb{T}$ is typically not equivalent (i.e., bisimilar [5]) to $\mathbb{D}$, but rather an over-approximation of $\mathbb{D}$, i.e., it may contain behaviors that do not exist in $\mathbb{D}$. Once $\mathbb{T}$ is constructed, a typical model checking procedure can be employed to check whether $\mathbb{T}$ satisfies a given LTL$_{\setminus\bigcirc}$ specification [1], [2]. Since $\mathbb{T}$ is an over-approximation of $\mathbb{D}$, if $\mathbb{T}$ satisfies $\mathbb{D}$, then we can conclude that $\mathbb{D}$ also satisfies φ. However, unless $\mathbb{T}$ is equivalent to $\mathbb{D}$, no conclusion about the correctness of $\mathbb{D}$ can be made otherwise. Hence, as our approach is not complete, the approaches based on explicit discretization of the dynamics are typically not complete, except for certain simple dynamics that allows $\mathbb{T}$ to be constructed such that it is equivalent to $\mathbb{D}$ [7].

Barrier certificates can also be utilized in these alternative approaches, particularly in the construction of $\mathbb{T}$. For example, we can construct $\mathbb{T}$ with $|2^{\Pi}|$ states where each state in $\mathbb{T}$ captures the states in $\mathbb{D}$ that satisfy the corresponding atomic propositions. Lemma 1 can be applied to remove transitions between states of $\mathbb{T}$ that cannot exist in $\mathbb{D}$. The computational complexity of this procedure may seem to be less than ours. However, even if computing barrier certificates can be automated based on Lemma 7, in practice, solving the sum-of-squares problem in Lemma 7 often requires some human guidance, particularly in selecting proper degrees of polynomials. Since $\mathbb{T}$ contains $|2^{\Pi}|$ states, $|2^{\Pi}|^2$ sum-of-squares problems need to be checked. In our approach, $|2^{\Pi}|^2$ transitions also need to be checked in the worst case. In practice though, the subpaths of length 3 considered in Step (3)-(b) of the LTL$_{\setminus\bigcirc}$ verification procedure often do not include all the $|2^{\Pi}|^2$ transitions. As a result, our approach allows to solve only the sum-of-squares problems that correspond to transitions that need to be checked based on these length 3 subpaths. In the example presented in Section IX, we consider the case where $|\Pi| = 3$; hence, $|2^{\Pi}| = 8$. Solving this problem using the alternative approaches requires considering 64 transitions whereas we show in Section IX that only 2 sum-of-squares problems need to be solved using our approach.

The approaches based on explicit discretization described above possibly lead to more conservative results than our approach because they typically utilize only Corollary 1 whereas both Corollary 1 and Corollary 2 can be applied in our approach. Consider, for example, a simple NBA $\mathcal{A}_{\neg\varphi}$ shown in Figure 4. Suppose no barrier certificates (see Lemma 1) can be found for the absence of trajectories starting from $[\![a_0]\!]$ and reaching $[\![a_1]\!]$ without leaving $[\![a_0]\!] \cup [\![a_1]\!]$, trajectories starting from $[\![a_1]\!]$ and reaching $[\![a_2]\!]$ without leaving $[\![a_1]\!] \cup [\![a_2]\!]$ and trajectories starting from $[\![a_2]\!]$ and reaching $[\![a_3]\!]$ without leaving $[\![a_2]\!] \cup [\![a_3]\!]$. In this case, a finite state abstraction of the dynamical system contains the transitions from a_0 to a_1, from a_1 to a_2, from a_2 to a_3 and from a_3 to a_3, leading to the conclusion that the correctness of the system cannot be verified. Further suppose that a barrier certificate can be found for the absence of trajectories starting from $[\![a_0]\!]$ and reaching $[\![a_2]\!]$ without

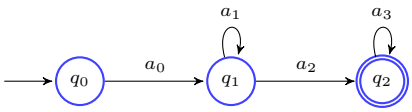

Fig. 4. A simple NBA $\mathcal{A}_{\neg\varphi}$ used in the discussion regarding the conservatism of approaches based on explicit discretization of dynamics compared to our approach. An arrow without a source points to an initial state. An accepting state is drawn with a double circle.

leaving $[\![a_0]\!] \cup [\![a_1]\!] \cup [\![a_2]\!]$.[3] This information cannot be utilized in the approaches based on explicit discretization of dynamics. With our approach, Corollary 2 can be used to conclude that the system is actually correct.

The conservatism of the approaches based on explicit discretization is often reduced by refining the state space partition based on the dynamics, resulting in larger abstract finite state systems [33]. As a result, these approaches face a combinatorial blow up in the size of the underlying discrete abstractions, commonly known as the state explosion problem.

D. Possible Extensions and Future Work

Throughout the paper, we consider a continuous vector field to ensure that x is sufficiently smooth, as required by Lemma 1 and Lemma 6, partly for ease of presentation. The approach presented in this paper, however, can potentially be extended to handle more general dynamics. For example, barrier certificates for safety verification of hybrid systems [9] can be utilized to extend Lemma 1 to handle hybrid systems. Such certificates, together with additional conditions to handle discrete jumps in Lemma 6, allow an extension of our approach to hybrid systems. Stochastic systems can potentially be handled using a similar idea. Such an extension is subject to future work.

Based on Proposition 1, we only consider subpaths of length 3. This restriction is due to the property that for any path π from q to q', there exists a path in $\mathcal{P}(q, q')$ whose all subpaths of length 3 can be extended in a simple way (by including possibly consecutive state repetitions) to be subpaths of π. However, this property may not necessarily hold for longer subpaths. For example, consider a graph $\mathcal{G}$ with $V^{\mathcal{G}} = \{q_0, q_1, q_2, q_3, q_4\}$ and $E^{\mathcal{G}} = \{(q_0, q_1), (q_1, q_2), (q_2, q_3), (q_3, q_1), (q_2, q_4)\}$. In this case, $\mathcal{P}(q_0, q_4) = \{q_0q_1q_2q_4\}$. Consider a path $\pi = q_0q_1q_2q_3q_1q_2q_4$. There does not exist any path in $\mathcal{P}(q_0, q_4)$ whose all subpaths of length greater than 3 can be extended only by including possibly consecutive state repetitions to be subpaths of π. It is possible to consider longer subpaths, provided that other ways of "extending" a subpath or other finite representative set of paths than those without any repeated edges or consecutive repetitions of states are considered. Note also that it is not useful to consider subpaths of length shorter than 3 since invalidating those subpaths requires proving that no trajectory can reach a certain region, say $\tilde{X}$, no matter where it starts. Such a condition cannot be verified since a trajectory that starts in $\tilde{X}$ always reaches $\tilde{X}$.

[3] See, for example, Figure 2. In this case, we can enlarge $\mathcal{X}_1$ such that there are trajectories starting from $\mathcal{X}_3$ and reaching $\mathcal{X}_1$ without leaving $\mathcal{X}_1 \cup \mathcal{X}_3$ and there are trajectories starting from $\mathcal{X}_1$ and reaching $\mathcal{X}_2$ without leaving $\mathcal{X}_1 \cup \mathcal{X}_2$. However, there are no trajectories starting from $\mathcal{X}_3$ and reaching $\mathcal{X}_1$ without leaving $\mathcal{X}_1 \cup \mathcal{X}_3$.

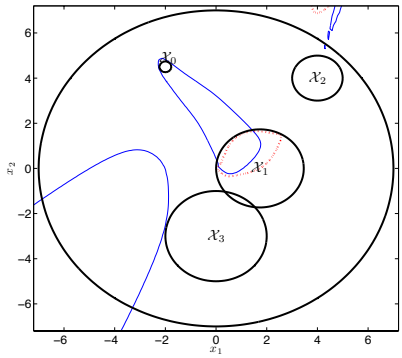

Fig. 5. The zero level sets of B (light solid blue curves) and $\frac{\partial B}{\partial x}(x)f(x)$ (dotted red curves) for π_1 where $\mathcal{Y}_0 = \mathcal{X}_0$, $\mathcal{Y}_1 = \mathcal{X}_2$ and $\mathcal{Y} = \mathcal{X} \setminus \mathcal{X}_1$.

Including longer subpaths helps reduce the conservatism of our approach. As the length of subpaths approaches infinity, we recover the set Ω, not only a set of its subpaths. An example similar to that provided in Section VIII-C can be constructed to show that considering longer subpaths could help reduce the conservatism of our approach. However, including longer subpaths results in increasing computational complexity.

IX. Example

Consider the problem defined in Example 1. As shown in Example 4, Algorithm V yields $\mathcal{P}^{cyc}(q_4) = \{q_4\}$ and $\mathcal{P}^{path}(q_4) = \{\pi_1, \pi_2, \pi_3\}$ where

$$\pi_1 = q_0q_1q_4, \quad \pi_2 = q_0q_2q_3q_4, \quad \pi_3 = q_0q_3q_4.$$

Since $\mathcal{P}^{cyc}(q_4)$ only contains one path $p = q_4$ and $\mathcal{PF}^3(p) = \emptyset$, conditions (1) of Proposition 1 cannot be satisfied. Hence, we consider condition (2), which requires checking all paths in $\mathcal{P}^{path}(q_4)$.

First, consider $\pi_1 = q_0q_1q_4$. In this case, we get $\mathcal{PF}^3(\pi_1) = \{\pi_1\}$. In addition, $\mathcal{ST}(\pi_1) = \{a_0a_1 \mid p_0 \in a_0, p_2 \in a_1\}$. Since $\overline{\mathcal{X}_0} \cap \overline{\mathcal{X}_2} = \emptyset$, we can conclude, using Lemma 6, that no finite string in $\mathcal{ST}(\pi_1)$ can be a substring of any word in $Trace(\mathbb{D})$. Since $(q_1, q_1) \in E^{\mathcal{G}}$, we also need to consider all finite strings in $\mathcal{ST}(q_0q_1q_1^+q_4) = \{a_0\tilde{a}_0 \dots \tilde{a}_k a_1 \mid k \in \mathbb{N}, p_0 \in a_0, p_1 \notin \tilde{a}_0, \dots, \tilde{a}_k, p_2 \in a_1\}$. Let $\mathcal{Y}_0 = \mathcal{X}_0$, $\tilde{Y} = \mathcal{X} \setminus \mathcal{X}_1$, $\mathcal{Y}_1 = \mathcal{X}_2$ and $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1 \cup \tilde{\mathcal{Y}} = \mathcal{X} \setminus \mathcal{X}_1$. Using SOSOPT, a polynomial B of degree 10, a constant $\epsilon > 0$ and the corresponding sum-of-squares polynomials $s_0(x), \dots, s_3(x)$ that make (9)-(11) sum-of-squares polynomials can be computed. Thus, we can conclude, using Corollary 2, that no finite string in $\mathcal{ST}(q_0q_1q_1^+q_4)$ can be a substring of any word in $Trace(\mathbb{D})$. The zero level sets of B and $\frac{\partial B}{\partial x}(x)f(x)$ are depicted in Figure 5, showing that $B(x) \leq 0$ for all $x \in \mathcal{X}_0$, $B(x) > 0$ for all $x \in \mathcal{X}_2$ and $\frac{\partial B}{\partial x}(x)f(x) \leq 0$ for all $x \in (\overline{\mathcal{X} \setminus \mathcal{X}_1}) \setminus \mathcal{X}_2$.

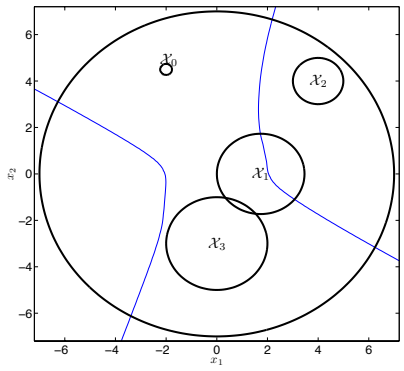

Fig. 6. The zero level set of B (light solid blue curves) for π_2' where $\mathcal{Y}_0 = \mathcal{X}_2$, $\mathcal{Y}_1 = \mathcal{X}_3$ and $\mathcal{Y} = \mathcal{X}$.

Next, consider $\pi_2 = q_0q_2q_3q_4$. In this case, $\mathcal{PF}^3(\pi_2) = \{q_0q_2q_3, q_2q_3q_4\}$. Let $\pi_2' = q_2q_3q_4$. As for the case of π_1, we can conclude that no finite string in $\mathcal{ST}(\pi_2')$ can be a substring of any word in $Trace(\mathbb{D})$ because $\overline{\mathcal{X}_2} \cap \overline{\mathcal{X}_3} = \emptyset$. Furthermore, for $\mathcal{ST}(q_2q_3q_3^+q_4) = \{a_0\tilde{a}_0 \dots \tilde{a}_k a_1 \mid k \in \mathbb{N}, p_2 \in a_0, p_3 \in a_1\}$, we let $\mathcal{Y}_0 = \mathcal{X}_2$, $\tilde{Y} = \mathcal{X}$, $\mathcal{Y}_1 = \mathcal{X}_3$ and $\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1 \cup \tilde{\mathcal{Y}} = \mathcal{X}$. SOSOPT generates a polynomial B of degree 8, a constant $\epsilon > 0$ and the corresponding sum-of-squares polynomials $s_0, \dots, s_3$ that make (9)-(11) sum-of-squares polynomials, ensuring that any trajectory of (6) that starts in $\mathcal{X}_2$ cannot reach $\mathcal{X}_3$ without leaving $\overline{\mathcal{X}}$. Thus, we can conclude, using Corollary 2, that no finite string in $\mathcal{ST}(q_2q_3q_3^+q_4)$ can be a substring of any word in $Trace(\mathbb{D})$. The zero level set of B is depicted in Figure 6, showing that $B(x) \leq 0$ for all $x \in \mathcal{X}_2$, $B(x) > 0$ for all $x \in \mathcal{X}_3$. Since $\frac{\partial B}{\partial x}(x)f(x) < 0$ for all $x \in \mathcal{X}$, the zero level set of $\frac{\partial B}{\partial x}(x)f(x)$ is not shown.

Finally, consider $\pi_3 = q_0q_3q_4$. In this case, $\mathcal{PF}^3(\pi_3) = \{\pi_3\}$. Furthermore, $\mathcal{ST}(\pi_3) = \mathcal{ST}(\pi_2')$ and $\mathcal{ST}(q_0q_3q_3^+q_4) = \mathcal{ST}(q_2q_3q_3^+q_4)$. Thus, we can use the results from π_2' to conclude that no finite string in $\mathcal{ST}(\pi_3) \cup \mathcal{ST}(q_0q_3q_3^+q_4)$ can be a substring of any word in $Trace(\mathbb{D})$.

At this point, we have checked all the paths in $\mathcal{P}^{path}(q_4)$ to conclude that condition (2) of Proposition 1 is satisfied. Thus, we can conclude that $\mathbb{D}$ satisfies φ.

X. Conclusions

An approach for computational verification of (possibly nonlinear) dynamical systems evolving over continuous state spaces subject to temporal logic specifications is presented. Typically, such verification requires checking the emptiness of the intersection of two sets, the set of all the possible behaviors of the system and the set of all the possible incorrect behaviors, both of which are potentially infinite, making the verification task challenging (if not impractical). In order to deal with these infinite sets, we propose a set of strings that, based on automata theory, can be used to represent the set of all the possible incorrect behaviors. Our approach then relies on constructing barrier certificates to ensure that each string in this set cannot be generated by any trajectory of the system. This integration of automata-based verification and barrier certificates allows us to avoid computing an explicit finite state abstraction of the continuous state space based on the underlying dynamics as commonly done in literature. Future work includes extending the presented approach to handle more general dynamics and attacking various sources of conservatism as discussed in the paper.

Acknowledgments

This work was supported in part by the AFOSR (FA9550-12-1-0302) and ONR (N00014-13-1-0778). The authors gratefully acknowledge Richard Murray for inspiring discussions.

References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[2] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[3] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Hybrid Systems: Computation and Control*, 2003, pp. 498–513.

[4] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.

[5] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[6] A. Girard and G. J. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 782–798, 2007.

[7] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *Journal of Computer and System Sciences*, vol. 57, pp. 94–124, 1998.

[8] R. Vinter, "A characterization of the reachable set for nonlinear control systems," *SIAM Journal on Control and Optimization*, vol. 18, no. 6, pp. 599–610, 1980.

[9] S. Prajna, "Optimization-based methods for nonlinear and hybrid systems verification," Ph.D. Dissertation, California Institute of Technology, 2005.

[10] P. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. Dissertation, California Institute of Technology, 2000, available at `http://thesis.library.caltech.edu/1647/`.

[11] U. Topcu, "Quantitative local analysis of nonlinear systems," Ph.D. Dissertation, UC, Berkeley, July 2008, available at `http://jagger.me.berkeley.edu/~utopcu/dissertation`.

[12] S. Prajna, A. Papachristodoulou, and F. Wu, "Nonlinear control synthesis by sum of squares optimization: A lyapunov-based approach," in *Asian Control Conference*, 2004, pp. 157–165.

[13] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. Packard, "Some controls applications of sum of squares programming," in *Conference Decision and Control*, vol. 5, 2003, pp. 4676–4681.

[14] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, pp. 2669 – 2675, 2008.

[15] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *Int. J. Rob. Res.*, vol. 29, no. 8, pp. 1038–1052, 2010.

[16] A. Galton, Ed., *Temporal Logics and Their Applications*. San Diego, CA: Academic Press Professional, Inc., 1987.

[17] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[18] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[19] A. He, J. Wu, and L. Li, "An efficient algorithm for transforming LTL formula to Büchi automaton," *Intelligent Computation Technology and Automation, International Conference on*, vol. 1, pp. 1215–1219, 2008.

[20] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*. London, UK: Springer-Verlag, 2001, pp. 53–65.
[21] D. Oddoux and P. Gastin, "LTL2BA : fast translation from LTL formulae to Büchi automata, version 0.2.2 beta," http://www.lsv.ens-cachan.fr/$\sim$gastin/ltl2ba/.
[22] G. J. Holzmann, "SPIN model checker," http://spinroot.com/spin/.
[23] M. Rönkkö, H. Tauriainen, and M. Mäkelä, "LBT: LTL to Büchi conversion," http://www.tcs.hut.fi/Software/maria/tools/lbt/.
[24] H. K. Khalil, *Nonlinear Systems*. Prentice-Hall, 1996.
[25] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.
[26] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Hybrid Systems: Computation and Control*, ser. LNCS, R. Alur and G. J. Pappas, Eds., vol. 2993. Springer, 2004, pp. 477–492.
[27] S. Prajna and A. Rantzer, "Primal-dual tests for safety and reachability," in *Hybrid Systems: Computation and Control*, ser. LNCS, M. Morari and L. Thiele, Eds., vol. 3414. Springer, 2005, pp. 542–556.
[28] G. Stengle, "A nullstellensatz and a positivstellensatz in semialgebraic geometry," *Mathematische Annalen*, vol. 207, no. 2, pp. 87–97, 1974.
[29] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 796–817, 2001.
[30] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, "Introducing SOSTOOLS: A general purpose sum of squares programming solver," in *Proceedings of the 41st IEEE Conf. on Decision and Control*, 2002, pp. 741–746.
[31] P. Seiler, "SOSOPT: A toolbox for polynomial optimization," 2013, arXiv:1308.1889.
[32] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theoretical Computer Science*, vol. 363, no. 2, pp. 182–195, 2006.
[33] B. Yordanov and C. Belta, "Formal analysis of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2834–2840, 2010.