Robustly Complete Reach-and-Stay Control Synthesis for Switched Systems via Interval Analysis

Yinan Li and Jun Liu

Abstract— **This paper proposes a formal synthesis algorithm for discrete-time switched systems with respect to reach-and-stay specifications. Fundamental to the proposed method is a fixed-point algorithm characterizing the initial states satisfying reach-and-stay specifications for continuous-state systems. Based on the interval branch-and-bound scheme, the original continuous state space is adaptively partitioned into a finite number of cells according to the given specification and system dynamics during the fixed-point iterations. Valid switching modes are recorded and a partition-based switching strategy can be extracted immediately after the algorithm terminates. In contrast with most of the abstraction-based methods, the proposed algorithm is guaranteed to return a switching strategy after a finite number of iterations, provided that the specification is robustly realizable. As illustrated in the numerical example, the adaptive partitioning framework effectively reduces the size of the finite partition, which offers a considerable advantage over abstraction-based methods that use a uniform partition.**

I. INTRODUCTION

The study of reachability and invariance control problems lies in the center of control theory and engineering. It is concerned with finding control signals that steer the system state into a specified target set and maintain the state in the target set afterwards. In practical applications, such as thermostat control, the outputs of feedback control systems are often regulated at a setpoint in the presence of disturbances. Driven by such applications, extensive research efforts have been put on reachability and invariance control for continuous physical systems based on set-theoretic methods and optimal control [1], [2], [3], [4].

In this paper, we restrict our attention to switched systems, which is an important class of hybrid systems. They are composed of multiple modes where the system state evolves continuously and the mode to be activated at a specific time is determined by a discrete variable. Such systems can be found in various applications, such as electrical power converters [5], robot motion planning [6], and flight management [7]. We are interested in the reach-and-stay objective: seeking a set of initial states and a switching strategy such that any state in the initial set can be controlled to reach and stay in a given target set of states. Such a set of initial conditions is called a *winning set*.

Abstraction-based (or symbolic) control synthesis can be used to solve the reach-and-stay control problem [8]. Since model checking and formal synthesis algorithms operate on finite-state systems, e.g., finite transition systems and automata [9], the first and foremost step is to construct a finite-state approximation for the original infinite-state system dynamics. Such an approximation is referred to as an abstraction or a symbolic model and is usually constructed by uniformly discretizing the state space. A control strategy is synthesized over the finite abstraction and refined to control the continuous system. Intuitively, it is favorable to construct (approximately) bisimilar models, which are (approximately) equivalent to the original systems. Systems known to have bisimilar models are limitted to controllable linear systems [10] and incremental stability is needed to construct an approximately bisimilar model [11]. Without these assumptions, over-approximations [12] or similar models [13] can be used to design provably correct control strategies, but they do not guarantee a feasible control strategy because spurious transitions are introduced by using over-approximations. The recent work [14] shows that both sound and approximately complete robust abstractions exist for discrete-time nonlinear systems without stability assumptions and can be obtained using sufficiently small discretization parameters. Using small grid sizes, however, is easy to render the computation of abstractions, as well as synthesis, intractable, because of the size of the abstraction scales exponentially with the dimension of the system.

Although fully automated, abstraction-based control synthesis is indirect in the sense that construction of abstractions and control synthesis are separated. As a result, the quality of abstractions has an impact on realizability of control synthesis. Abstraction-refinement loops [15] can be introduced to refine the abstractions when a given specification is not realizable; however, such loops are often not guaranteed to terminate for continuous-state systems. Unlike abstraction-based methods, we propose a synthesis algorithm directly over the original switched system without any abstraction stage. It is based on a fixed-point iteration defined over the continuous state space. We show that this fixed-point characterization can be used to determine whether a reach-and-stay specification is robustly realizable for a switched system. In other words, the proposed direct synthesis algorithm is finitely terminating and generates a partition-based switching strategy provided that the given specification can be satisfied under perturbation. The essential technique for the design and implementation of the proposed algorithm is interval arithmetic computation [16], which is often used for rigorous and reliable analysis and computation.

In comparison with related work in the literature, we highlight the following main contributions. Firstly, an automated reach-and-stay synthesis algorithm is proposed for switched

Y. Li and J. Liu are with the Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada. yinan.li@uwaterloo.ca, j.liu@uwaterloo.ca

systems. The proposed algorithm is proved to be robustly complete for switched systems in the sense that it returns a switching strategy whenever the given specification is realizable for the systems with some degree of uncertainties. It extends our previous results on robustly controlled invariance [17] to more general specifications involves both invariance and reachability. A related result in the context of abstraction-based control is presented in [14] with focus on constructing approximately complete robust abstractions, where details on synthesis and the corresponding computational complexity were not discussed. In another line of related work reported in [18], bounded reachability of hybrid systems is shown to be robustly decidable. The major difference between their work and the current paper is that we do not assume a finite reachability horizon as in [18]. In addition, our objective is control synthesis while they focus on reachability analysis.

Secondly, the proposed direct control synthesis algorithm is of higher efficiency than abstraction-based methods with uniform grids. The size of the non-uniform partition and control strategy generated by the proposed method is smaller than that of a uniform abstraction with the same discretization precision. The rationale behind it is that the infinite state space is adaptively partitioned into non-uniform intervals with respect to both the specification and system dynamics by applying the interval branch-and-bound scheme in our algorithm. As another factor contributing to the higher efficiency, we show that memoryless switching strategies can be directly extracted, and no additional synthesis stage is needed. The advantage of using non-uniform partitions is especially pronounced when the target invariance region demands a high control precision, as illustrated in the example in section V.

The structure of the current paper is as follows. In section II, we provide preliminaries on linear temporal logic and formulate the reach-and-stay control problem. In Section III, we describe a fixed-point characterization of the winning set of a given reach-and-stay specification for continuous-state systems. Section IV presents a direct synthesis algorithm based on interval computation and proves the completeness of the resulting switching strategies under some robustness condition. In Section V, we illustrate the efficiency and effectiveness of the proposed method by an inverted pendulum control example.

Notation: Let $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{R}^n$ be the set of all integers, reals, and n-dimensional real vectors, respectively; the subscript ≥ 0 (> 0) in a set denotes the set of non-negative (positive) elements in the set, e.g. $\mathbb{Z}_{\geq 0}$ is the set of non-negative integers; given $A \subseteq \mathbb{R}^n$, the interior and closure of A are denoted by $\text{int}(A)$ and $\text{cl}(A)$, respectively; a compact set A is called *full* if $A = \text{cl}(\text{int}(A))$; given two sets $A, B \subseteq \mathbb{R}^n$, $B \setminus A := \{x \in B \,|\, x \notin A\}$; the Pontryagin difference is defined as $A \ominus B := \{c \in \mathbb{R}^n \mid c + b \in A, \forall b \in B\}$; define $\mathcal{B}_r := \{y \in \mathbb{R}^n \mid |y| \leq r\}$, where $|\cdot|$ is the infinity norm in $\mathbb{R}^n$; an interval vector (interval for short) in $\mathbb{R}^n$ is denoted by $[x]$, where $[x] := [x_1] \times \cdots \times [x_n] \subseteq \mathbb{R}^n$, $[x_i] = [\underline{x}_i, \overline{x}_i] \subseteq \mathbb{R}$ for $i = 1, \cdots, n$, $\underline{x}_i$ represents the infimum of $[x_i]$, and $\overline{x}_i$ the supremum; we also write $[x] = [\underline{x}, \overline{x}]$, where $\underline{x} = [\underline{x}_1, \cdots, \underline{x}_n]^T$ and $\overline{x} = [\overline{x}_1, \cdots, \overline{x}_n]^T$; the width of the interval $[x]$ is defined as $w([x]) := \max_{1 \leq i \leq n} \{\overline{x_i} - \underline{x_i}\}$; the set of all intervals in $\mathbb{R}^n$ is denoted by $\mathbb{IR}^n$; given $X, Y \subseteq \mathbb{IR}^n$, $X \subseteq Y$ denotes $\bigcup_{[x] \in X} [x] \subseteq \bigcup_{[y] \in Y} [y]$; given two functions f and g, the composite function $g \circ f(\cdot) := g(f(\cdot))$.

II. PROBLEM FORMULATION

In this section, we introduce linear temporal logic for the formal definition of reach-and-stay objectives and formulate the control synthesis problem.

A. Discrete-time switched systems

We consider discrete-time switched systems of the form:

$$x_{k+1} = f_{p_k}(x_k), \tag{1}$$

where $k \in \mathbb{Z}_{\geq 0}$ is a discrete time instant, $\mathcal{M}$ denotes the finite set of modes, $x_k \in \mathbb{R}^n$ and $p_k \in \mathcal{M}$ is the system state and mode at time k, respectively, and $\{f_p\}_{p \in \mathcal{M}} : \mathbb{R}^n \to \mathbb{R}^n$ is a set of continuous functions governing system evolution.

Any infinite sequence taking values in $\mathcal{M}$ defines a *switching signal* for system (1). We denote a particular switching signal by $\mathbf{p} := \{p_k\}_{k=0}^{\infty}$, where $p_k \in \mathcal{M}$ for all $k \geq 0$. Given a switching signal $\mathbf{p}$ and an initial state $x_0 \in \mathbb{R}^n$, the solution of system (1) is the unique sequence $\mathbf{x} := \{x_k\}_{k=0}^{\infty}$ in $\mathbb{R}^n$ such that (1) is satisfied.

B. The reach-and-stay control problem

For the sake of formal synthesis, we write reach-and-stay objectives in the form of Linear temporal logic. Linear temporal logic (LTL) is a formalism defined over a set of *atomic propositions*, which are true or false statements. We denote the set of atomic propositions by AP. An LTL formula consists of propositional logic operators (e.g., **true**, **false**, *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$) and *implication* ($\rightarrow$)), and temporal operators (e.g., *next* ($\bigcirc$), *until* (**U**), *always* ($\Box$), *eventually* ($\Diamond$). The syntax of LTL in Backers Naur form is

$$\varphi ::= p \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \mathbf{U} \varphi \,|\, \bigcirc \varphi,$$

where p is an atomic proposition. Details regarding LTL can be found in [9].

To interpret an LTL formula over a system solution, we define a *labeling function* $L : \mathbb{R}^n \to 2^{AP}$, which associates a set of atomic propositions to every state in $\mathbb{R}^n$. The *trace* of a solution $\mathbf{x}$ is defined as $\text{Trace}(\mathbf{x}) := \{L(x_i)\}_{i=0}^{\infty}$.

Written as an LTL formula, the reach-and-stay objective is in the following form:

$$\varphi = \Box a_s \wedge \Diamond \Box a_g \tag{2}$$

where a_s and a_g atomic propositions. The set of states labeled by a_s and a_g are denoted by $[\![a_s]\!]$ and $[\![a_g]\!]$, respectively, i.e., $[\![a_s]\!] = \{x \in \mathbb{R}^n \mid a_s \in L(x)\}$ and $[\![a_g]\!] = \{x \in \mathbb{R}^n \mid a_g \in L(x)\}$.

The first term $\Box a_s$ poses a safety specification, under which the system state has to stay inside the "safe" area labeled by the atomic proposition a_s. The second term $\Diamond \Box a_g$ represents the main reach-and-stay objective, which requires the existence of a time $j > 0$ at which $[\![a_g]\!]$ is reached

and maintained for all time $i \geq j$. Note that the existence quantification poses no upper bound to the time j, which means an infinite reachability horizon. A typical example of such specifications is practical stabilization, in which the system state is to be controlled into a small neighborhood around a setpoint.

The purpose of this paper is to design a switching strategy such that the resulting solutions of (1) satisfy (2) if feasible. Prior to the formal problem statement, we provide the following definition.

Definition 1: A *(memoryless) switching strategy* of system (1) is a function

$$\kappa : \mathbb{R}^n \to 2^{\mathcal{M}}. \tag{3}$$

A (state-dependent) switching signal $\mathbf{p} = \{p_k\}_{k=0}^{\infty}$ is said to *conform to* a switching strategy κ, if

$$p_k \in \kappa(x_k), \quad \forall k \geq 0, \tag{4}$$

where $\{x_k\}_{k=0}^{\infty}$ is the resulting solution of (1).

If there exists an initial condition $x_0 \in S$ and a switching strategy κ such that, for any switching signal that conforms to κ, the resulting system solution satisfies a given LTL formula φ, we say φ is *realizable* for system (1). Such a switching strategy κ is called a *realization* of φ for system (1). The *winning set* of φ is the set of all initial states from which φ is realizable, written as $\text{Win}(\varphi)$. Thus, if $\text{Win}(\varphi) \neq \varnothing$, then φ can be realized for system (1).

Now we formulate the control synthesis problem: consider system (1) and an LTL formula φ in the form of (2). Determine whether φ is realizable for system (1) and design a switching strategy such that the resulting solutions satisfy φ if φ is realizable.

III. FIXED-POINT CHARACTERIZATION OF REALIZABILITY

This section shows that the realizability of an LTL formula φ in the form of (2) for system (1) can be determined through a fixed-point algorithm performed over the continuous state space. The winning set of φ is characterized by the fixed point (set) returned by the algorithm. Considering that the system state space is always bounded in real applications, it is fair to assume the compactness of the sets of states labeled by a_g and a_s.

The following definition is fundamental to the fixed-point characterization.

Definition 2: Given a set $X \subseteq \mathbb{R}^n$, the *predecessor* of X with respect to system (1) is a set of states defined by

$$\text{Pre}(X) := \{x \in \mathbb{R}^n \mid \exists p \in \mathcal{M} \text{ s.t. } f_p(x) \in X\}. \tag{5}$$

We can also interpret $\text{Pre}(\cdot)$ as a map between subsets of $\mathbb{R}^n$, and as for general nonlinear systems [19], [4], the following properties can be derived straightforwardly for the switched systems.

Proposition 1: Let $A, B \subseteq \mathbb{R}^n$. Then (i) if A is closed, $\text{Pre}(A)$ is closed; (ii) if $A \subseteq B$, then $\text{Pre}(A) \subseteq \text{Pre}(B)$.

Based on computation of predecessors, Algorithm 1 returns the winning set defined in our control synthesis problem, as a fixed point of two nested while loops. We show in Proposition 2 that such a characterization is sound and complete.

Algorithm 1 Computation of $\text{Win}(\Box a_s \wedge \Diamond\Box a_g)$

Require: $[\![a_s]\!], [\![a_g]\!]$
1: $G \leftarrow [\![a_s]\!] \cap [\![a_g]\!]$
2: $Y \leftarrow [\![a_s]\!], \widetilde{Y} \leftarrow \emptyset$
3: **while** $\widetilde{Y} \neq Y$ **do**
4: $\quad Y \leftarrow \widetilde{Y}$
5: $\quad Z \leftarrow [\![a_s]\!] \cap \text{Pre}(Y)$
6: $\quad X \leftarrow \emptyset, \widetilde{X} \leftarrow G \cup Z$
7: $\quad$ **while** $\widetilde{X} \neq X$ **do**
8: $\quad\quad X \leftarrow \widetilde{X}$
9: $\quad\quad \widetilde{X} \leftarrow Z \cup (X \cap \text{Pre}(X))$
10: $\quad$ **end while**
11: $\quad \widetilde{Y} \leftarrow X$
12: **end while**
13: **return** Y

Proposition 2: Let $[\![a_g]\!], [\![a_s]\!]$ be nonempty compact subsets of $\mathbb{R}^n$. Then there exists a fixed point of Algorithm 1, denoted by Y^{∞}. Furthermore, $\text{Win}(\Box a_s \wedge \Diamond\Box a_g) = Y^{\infty}$.

Proof: Let $\{Y_k\}_{k=0}^{\infty}$ and $\{Z_k\}_{k=0}^{\infty}$ be the set sequences generated by the outer loop, where $Z_k = [\![a_s]\!] \cap \text{Pre}(Y_k)$. In kth iteration, the set sequence $\{X_k^j\}_{j=0}^{\infty}$ is determined by

$$\begin{cases} X_k^0 & = Z_k \cup G, \\ X_k^{j+1} & = Z_k \cup (X_k^j \cap \text{Pre}(X_k^j)). \end{cases} \tag{6}$$

Then $X_k^1 = Z_k \cup (X_k^0 \cap \text{Pre}(X_k^0)) \subseteq Z_k \cup G = X_k^0$. Assume that $X_k^{j+1} \subseteq X_k^j$. By Proposition 1 (ii), $\text{Pre}(X_k^{j+1}) \subseteq \text{Pre}(X_k^j)$, and thus $X_k^{j+2} = Z_k \cup (X_k^{j+1} \cap \text{Pre}(X_k^{j+1})) \subseteq Z_k \cup (X_k^j \cap \text{Pre}(X_k^j)) = X_k^{j+1}$. Hence, $\{X_k^j\}$ is decreasing, and $Z_k \subseteq X_k^j \subseteq Z_k \cup G$ for all $j, k \in \mathbb{Z}_{\geq 0}$. By Tarski-Knaster fixed point theorem [20], if $Z_k \neq \emptyset$, then $X_k^{\infty} := \lim_{j\to\infty} X_k^j = \bigcap_{j\in\mathbb{Z}_{\geq 0}} X_k^j$ is a unique fixed point of the map defined by (6).

We first claim that $\{Y_k\}$ is an increasing sequence bounded by $[\![a_s]\!]$. This indicates the existence of a fixed point of Algorithm 1. For $k = 0$, $Z_0 = Y_0 = \emptyset$. For $k = 1$, $Y_1 = X_0^{\infty}$ and $Y_1 = Y_1 \cap \text{Pre}(Y_1) \subseteq [\![a_s]\!] \cap \text{Pre}(Y_1) = Z_1$. Hence, $Y_0 \subseteq Y_1 \subseteq Z_1$ and $Z_0 \subseteq Z_1$. Assume that $Y_{k-1} \subseteq Y_k \subseteq Z_k$, and $Z_{k-1} \subseteq Z_k$. Then $Y_{k+1} = X_k^{\infty} = Z_k \cup (X_k^{\infty} \cap \text{Pre}(X_k^{\infty})) \supseteq Y_k$ and $Z_k = [\![a_s]\!] \cap \text{Pre}(Y_k) \subseteq [\![a_s]\!] \cap \text{Pre}(Y_{k+1}) = Z_{k+1}$. Moreover,

$$\begin{aligned} Y_{k+1} &= Z_k \cup (Y_{k+1} \cap \text{Pre}(Y_{k+1})) \\ &\subseteq Z_{k+1} \cup (Y_{k+1} \cap \text{Pre}(Y_{k+1})) \\ &= ([\![a_s]\!] \cap \text{Pre}(Y_{k+1})) \cup (Y_{k+1} \cap \text{Pre}(Y_{k+1})) \\ &= [\![a_s]\!] \cap \text{Pre}(Y_{k+1}) = Z_{k+1}. \end{aligned}$$

Hence, $Y_k \subseteq Y_{k+1} \subseteq Z_{k+1}$, and $Z_k \subseteq Z_{k+1}$, which means that the claim holds and there exists a fixed point $Y^{\infty} := \lim_{k\to\infty} Y_k = \bigcup_{k\in\mathbb{Z}_{\geq 0}} Y_k = \lim_{k\to\infty} Z_k$.

Next, we show that $Y^{\infty} = \text{Win}(\varphi)$, where $\varphi = \Box a_s \wedge \Diamond\Box a_g$. We prove $Y^{\infty} \subseteq \text{Win}(\varphi)$ by induction. For the base

case $k = 1$, $Y_1 = G \cap \text{Pre}(Y_1)$ because Y_1 is a fixed point of the inner loop. It follows that Y_1 is the maximal controlled invariant set inside G by [2, Proposition 4] and thus $Y_1 \subseteq \text{Win}(\Box a_s \wedge \Box a_g)$. By definition, $Z_k = [\![a_s]\!] \cap \text{Pre}(Y_k)$. Then any state in $Z_k \subseteq [\![a_s]\!]$ can be controlled into Y_k in one step, and thus if $Y_k \subseteq \text{Win}(\varphi)$, $Z_k \subseteq \text{Win}(\varphi)$ as well. Hence for the base case, $Z_1 \subseteq \text{Win}(\varphi)$. Suppose that $Y_k \subseteq Z_k \subseteq \text{Win}(\varphi)$. Taking the fixed point of the iteration (6), any state in Y_{k+1} can be maintained inside $Z_k \cup G$, which implies $Y_{k+1} \subseteq Z_{k+1} \subseteq \text{Win}(\varphi)$. Since k is arbitrary, we have $Y^\infty \subseteq \text{Win}(\varphi)$.

To see $\text{Win}(\varphi) \subseteq Y^\infty$, we show that for all $y \notin Y^\infty$, $y \notin \text{Win}(\varphi)$. As a fixed point of the outer loop, any state in Y^∞ can stay inside Y^∞ for all time. Meanwhile, any state $y \notin Y^\infty$ can not be controlled inside Y^∞ at any time, because otherwise y will be collected in Y^∞ at some time step. We now only consider the case $(G \cap Y^\infty \neq \emptyset) \wedge (G \cap Y^\infty \neq G)$, since the claim holds with $G \subseteq Y^\infty$ and the previous arguments. According to the inner loop (6), Y^∞ is the maximal controlled invariant set contained in $G \cup Y^\infty$. Let $G_c := G \setminus Y^\infty$. Then for any state $x \in G_c$, x can neither stay inside G for all time nor controlled inside Y^∞. Hence, $x \notin \text{Win}(\varphi)$. For any state $y' \notin G \cup Y^\infty$, y can only be controlled into G_c if it is possible, which implies that $y' \notin \text{Win}(\varphi)$. Therefore, $\text{Win}(\varphi) \subseteq Y^\infty$, and this completes the proof. ∎

If the output of Algorithm 1 is nonempty, then φ is realizable for (1) and a switching strategy can be found. Otherwise φ is unrealizable.

Algorithm 1 has been shown to be sound and complete over finite game structures [21].

Proposition 2 spells out that the same algorithm is also sound and complete for continuous-state systems, even at the present of additive disturbances. To see this, let us consider the perturbed system

$$x_{k+1} = f_{p_k}(x_k) + d_k, \quad k \in \mathbb{Z}_{\geq 0}, \tag{7}$$

where $d_k \in \mathcal{B}_\delta$, $\delta \in \mathbb{R}_{\geq 0}$, is a perturbation. The predecessor of a set X with respect to system (7) under any perturbation is defined as

$$\text{Pre}^\delta(X) := \{x \in \mathbb{R}^n \mid \forall d \in \mathcal{B}_\delta, \exists p \in \mathcal{M}, \text{ s.t. } f_p(x) + d \in X\}.$$

By the definition of Pontryagin difference, $\text{Pre}^\delta(X) = \text{Pre}(X \ominus \mathcal{B}_\delta)$. It follows that $\text{Pre}^\delta(X)$ is a subset of $\text{Pre}(X)$. Proposition 1 is preserved for the map $\text{Pre}^\delta(\cdot)$, because $X \ominus \mathcal{B}_\delta$ is closed (compact) provided that X is closed (compact) and $X \ominus \mathcal{B}_\delta \neq \emptyset$. Therefore, Proposition 2 for the unperturbed system (1) also holds for the perturbed system (7). Denote by $\text{Win}^\delta(\varphi)$ the winning set of an LTL formula φ for system (7). Replacing map $\text{Pre}(\cdot)$ by $\text{Pre}^\delta(\cdot)$, Algorithm 1 computes the winning set of the specification (2) for system (7), i.e., $\text{Win}^\delta(\Box a_s \wedge \Diamond \Box a_g)$.

The above algorithm, however, is impractical for actual computation because it might not terminate in a finite number of iterations and the computation of predecessors for nonlinear dynamics is numerically nontrivial.

IV. CONTROL SYNTHESIS ALGORITHM USING INTERVAL METHODS

As our main result, a fully automated algorithm is presented in this section for solving the reach-and-stay synthesis problem, which is built upon interval approximation of predecessors under nonlinear dynamics. The proposed algorithm is finitely terminating and generates switching strategies with some robustness guarantee.

A. Interval approximation of predecessors

Central to approximations of set images under the nonlinear map $\text{Pre}(\cdot)$ is the following definition.

Definition 3 ([16]): Consider a function $f : \mathbb{R}^n \to \mathbb{R}^m$. An interval function $[f] : \mathbb{IR}^n \to \mathbb{IR}^m$ is called a *convergent inclusion function* of f if the following conditions hold:

(i) $f([x]) \subseteq [f]([x])$ for all $[x] \in \mathbb{IR}^n$;
(ii) $\lim_{w([x]) \to 0} w([f]([x])) = 0$.

Such a convergent inclusion function is not unique for a given function f defined on $\mathbb{R}^n$. The natural inclusion function, which is obtained by applying interval operation rules directly to the same real-valued function, and mean-value inclusion function are usually used [16].

Let X and Y be subsets of $\mathbb{R}^n$ and represented by intervals or unions of intervals. Using branch-and-bound scheme, Algorithm 2 approximates the predecessor of Y that resides in set X, i.e., $X \cap \text{Pre}(Y)$, and the set approximation error is controlled by a parameter $\varepsilon > 0$.

Algorithm 2 Predecessor of Y bounded by X

procedure CPRED($[f_p]_{p \in \mathcal{M}}, X, Y, \varepsilon$)
  $K \leftarrow \emptyset$
  $\underline{X} \leftarrow \emptyset, \Delta X \leftarrow \emptyset, X_c \leftarrow \emptyset, List \leftarrow X$
  while $List \neq \emptyset$ **do**
    $[x] \leftarrow List.first$
    if $[f_p]([x]) \cap Y = \emptyset$ for all $p \in \mathcal{M}$ **then**
      $X_c \leftarrow X_c \cup [x]$
    else if $[f_p]([x]) \subseteq Y$ for some $p \in \mathcal{M}$ **then**
      $\underline{X} \leftarrow \underline{X} \cup [x]$
      $K \leftarrow K \cup ([x], p)$
    else
      if $w([x]) < \varepsilon$ **then**
        $\Delta X \leftarrow \Delta X \cup [x]$
      else
        $\{L[x], R[x]\} = Bisect([x])$
        $List.add(\{L[x], R[x]\})$
      end if
    end if
  end while
  return $K, \underline{X}, \Delta X, X_c$
end procedure

The intervals that entirely belong to $X \cap \text{Pre}(Y)$ are collected in $\underline{X}$ while those mapped outside of Y by $[f_p]$ for any $p \in \mathcal{M}$ are collected in X_c. If an interval $[x]$ with width

greater than ε can not be determined, then $[x]$ is bisected to

$$L[x] = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_j, (\underline{x}_j + \overline{x}_j)/2] \times \cdots \times [\underline{x}_n, \overline{x}_n],$$
$$R[x] = [\underline{x}_1, \overline{x}_1] \times \cdots \times [(\underline{x}_j + \overline{x}_j)/2, \overline{x}_j] \times \cdots \times [\underline{x}_n, \overline{x}_n],$$

where j is the dimension in which the box x attains its width. The list of undetermined intervals with width less than ε is denoted by ΔX. We call $\underline{X}$ the inner approximation and $\overline{X} := \underline{X} \cup \Delta X$ the outer approximation of $X \cap \text{Pre}(Y)$.

In addition, Algorithm 2 returns a set K containing pairs of intervals and their corresponding switching modes under which the intervals are mapped into Y completely in one step of time. We denote such a pair by $([x], \{p\})$, and the modes in $\{p\}$ are called *valid switching modes*. The inner-approximation of $\text{Pre}(Y)$ is called the interval part of K.

More generally, the input set Y of Algorithm 2 needs not to be intervals. It can be defined by equations or inequalities, e.g. $Y := \{y \in \mathbb{R}^n \mid g(y) \leq 0\}$, where $g : \mathbb{R}^n \to \mathbb{R}^m$. In this case, the condition $[f_p]([x]) \cap Y = \emptyset$ and $[f_p]([x]) \subseteq Y$ in Algorithm 2 are replaced by $[g \circ f_p]([x]) \subseteq [0, \infty]^m$ and $[g \circ f_p]([x]) \subseteq [-\infty, 0]^m$, respectively.

The parameter ε controls the minimum width of intervals for approximating $X \cap \text{Pre}(Y)$, which is called the precision control parameter. Proposition 3 below establishes the relation between ε and the set approximation error, which relies on Assumption 1 below.

Assumption 1: Let $D \subseteq \mathbb{R}^n$. There exists a constant $\rho > 0$ such that system (1) satisfies the following local Lipschitz condition for all $p \in \mathcal{M}$:

$$|f_p(x) - f_p(y)| \leq \rho |x - y|, \quad \forall x, y \in D. \tag{8}$$

If f_p is continuously differentiable on D for all $p \in \mathcal{M}$, and D is compact, then $\rho = \max_{x \in \overline{\text{co}}(D), p \in \mathcal{M}} \|J_x f_p\|$, where J_x is the Jacobian matrix at x, and $\|\cdot\|$ denotes the matrix operator norm. Based on (8) and Definition 3, it is always possible to choose an inclusion function $[f_p]$ for each f_p (e.g., mean-value form [16]) such that

$$w([f_p]([x])) \leq \rho w([x]), \quad \forall [x] \subseteq \Omega. \tag{9}$$

Proposition 3 ([17]): Let D be a subset of $\mathbb{R}^n$ and $Y, X \subseteq D$ be compact. Denote that $[K, \underline{Z}, \Delta Z, Z_c] = \text{CPRED}([f_p]_{p \in \mathcal{M}}, X, Y, \varepsilon)$. If Assumption 1 holds on D, and the interval functions $[f_p]$ ($p \in \mathcal{M}$) are chosen such that (9) is satisfied, then

$$X \cap \text{Pre}(Y \ominus \mathcal{B}_{\rho\varepsilon}) \subseteq \underline{Z} \subseteq X \cap \text{Pre}(Y).$$

Under more strict conditions and the set limit defined in [22, Definition 4.1], we can prove that the inner-approximation $\underline{X}$ converges to the exact set $X \cap \text{Pre}(Y)$ as ε goes to zero. This implies that predecessors can be approximated with arbitrary precision.

B. Interval-based control algorithm for robustly realizable LTL specifications

As we have shown in the previous section, the sets $[\![a_s]\!] \cap \text{Pre}(Y)$ and $X \cap \text{Pre}(X)$ in Algorithm 1 can be approximated by applying $\text{CPRED}([f_p]_{p \in \mathcal{M}}, [\![a_s]\!], Y, \varepsilon)$ and $\text{CPRED}([f_p]_{p \in \mathcal{M}}, X, X, \varepsilon)$, respectively. Meanwhile, the valid switching modes for the predecessors are recorded. Inner approximations are often used because, for the purpose of control synthesis, a switching strategy has to exist for all states in the approximated winning set.

We now present the following algorithm as an interval implementation of Algorithm 1. It returns a set K, which is a list of pairs representing a subset of the real winning set and the valid switching modes. The precision of such a subset is adjusted by the precision control parameter ε. A smaller ε produces a more precise approximation.

Algorithm 3 Approximation of $\text{Win}(\Box a_s \wedge \Diamond \Box a_g)$

Require: $[\![a_g]\!], [\![a_s]\!], [f_p]_{p \in \mathcal{M}}, \varepsilon$
1: $\underline{K} \leftarrow \emptyset$
2: $\widetilde{Y} \leftarrow \emptyset, Y \leftarrow [\![a_s]\!], Z \leftarrow \emptyset$
3: $G_1 \leftarrow [\![a_s]\!] \setminus [\![a_g]\!], G_2 \leftarrow [\![a_s]\!] \cap [\![a_g]\!]$
4: **while** $\widetilde{Y} \neq Y$ **do**
5: $\quad Y \leftarrow \widetilde{Y}$
6: $\quad [K_z, \underline{Z}, \Delta Z, Z_c] = \text{CPRED}([f_p]_{p \in \mathcal{M}}, G_1, Y, \varepsilon)$
7: $\quad K \leftarrow K \cup K_z$
8: $\quad Z \leftarrow Y \cup \underline{Z}$
9: $\quad X \leftarrow \emptyset, \widetilde{X} \leftarrow Z \cup G_2, V \leftarrow G_2, G_2 \leftarrow \emptyset$
10: $\quad$ **while** $\widetilde{X} \neq X$ **do**
11: $\quad\quad X \leftarrow \widetilde{X}$
12: $\quad\quad [\underline{K}_v, \underline{V}, \Delta V, V_c] = \text{CPRED}([f_p]_{p \in \mathcal{M}}, V, X, \varepsilon)$
13: $\quad\quad \widetilde{X} \leftarrow Z \cup \underline{V}$
14: $\quad\quad V \leftarrow \underline{V}$
15: $\quad\quad G_2 \leftarrow G_2 \cup \Delta V \cup V_c$
16: $\quad$ **end while**
17: $\quad \underline{K} \leftarrow K \cup K_v$
18: $\quad \widetilde{Y} \leftarrow X$
19: $\quad G_1 \leftarrow \Delta Z \cup Z_c$
20: **end while**
21: **return** K

A direct interval translation of line 5 and 9 of Algorithm 1 has to enumerate every intervals that are parts of the input set $[\![a_s]\!]$ and $G \cup Z$ of the procedure CPRED. The numbers of the intervals contained in these sets increase as the fixed-point iteration proceeds, yet some of these intervals have been verified to be in the winning set from previous iterations in the procedure CPRED. To reduce computational complexity, we use two additional sets G_1 and G_2 to keep tracking the intervals necessary to test. In Algorithm 3, the safe region $[\![a_s]\!]$ is initialized into two parts. The region inside $[\![a_g]\!]$ is denoted by G_2 while the rest of $[\![a_s]\!]$ is denoted by G_1. During each outer iteration, G_2 is refined and only holds the intervals that cannot be always controlled inside $[\![a_s]\!] \cap [\![a_g]\!]$ based on current Y. Similarly, G_1 retains the part that cannot reach Y. As a result, the regions represented by G_1 and G_2 are decreasing until they no longer change.

Practical control systems normally suffer from imperfections in multiple aspects of the feedback control scheme. Measurements are corrupted by noise. Delay happens in transferring measured data from sensors to controllers and also from controllers to plants. In digital control systems, numerical errors are hardly avoided as data is quantified

when passing through A/D and D/A modules and approximations are often used to perform control algorithms. Thus, the requirement that specifications can be robustly realized is meaningful in control design. Hence, we are concerned with the robust realizability of specifications.

Definition 4: An LTL formula φ is said to be δ-robustly realizable for system (1) if it is realizable for system (7). If $\delta > 0$, then φ is called robustly realizable for system (1).

In this context, we show in Theorem 1 that the approximation error can be tolerated by the robust realizability of the specification (2).

Theorem 1: Let D be a subset of $\mathbb{R}^n$ and $\varphi = \Box a_s \wedge \Diamond\Box a_g$, where $[\![a_s]\!], [\![a_g]\!] \subseteq D \subseteq \mathbb{R}^n$ are compact. Assume that φ is δ-robustly realizable for system (1) and Assumption 1 holds on D. Denote by Y^ε the interval part of the output of Algorithm 3 for a given $\varepsilon > 0$. Then Algorithm 3 terminates in a finite number of steps, and the following relation holds if $\rho\varepsilon \leq \delta$:

$$\mathrm{Win}^\delta(\varphi) \subseteq Y^\varepsilon \subseteq \mathrm{Win}(\varphi). \tag{10}$$

Proof: Denote by i the index of the outer while loop and j the one of the inner loop of Algorithm 3. Let Y_i, Z_i, X_i and V_i be Y, Z, X and V by the end of ith iteration, respectively. We use X_i^j and V_i^j to indicate X_i and V_i by the end of the jth inner iteration. Initially, $Z_0 = Y_0 = \emptyset$, and $V_0^0 = G := [\![a_s]\!] \cap [\![a_g]\!]$. For all $i, j \in \mathbb{Z}_{\geq 0}$, $V_i^j \subseteq G$.

We first show that these two nested loops terminate. For a fixed i, V_i^0 is compact, and the sequence $\{V_i^j\}$ is decreasing with respect to j. Under a given precision $\varepsilon > 0$, V_i^0 can only be partitioned to finite number of intervals. Then there must exist a positive integer N such that $V_i^N = \emptyset$ if $V_i^j \neq V_i^{j+1}$ for all $j \leq N-1$. Thus, the inner loop terminates eventually. Let V_i^ε denote V_i after the inner loop terminates at the ith iteration. By line 6, $G_{1,i} = \underline{Z}_{i+1} \cup \Delta Z_{i+1} \cup Z_{c,i+1}$ but $G_{1,i+1} = \Delta Z_{i+1} \cup Z_{c,i+1} \subseteq G_{1,i}$, which means that $G_{1,i}$ is non-increasing. If at some iteration k, $G_{1,k} = G_{1,k+1}$, then $\underline{Z}_k = \emptyset$. It follows that $Z_k = Z_{k+1}$, $G_{2,k} = G_{2,k+1}$ and finally $Y_k = Y_{k+1}$. Otherwise G_1 is strictly decreasing. Since $[\![a_s]\!] \setminus [\![a_g]\!]$ only contains finite number of intervals, there exist a positive integer M such that $G_{1,M} = \emptyset$. Hence, the outer loop also terminates, which implies Algorithm 3 terminates in finite number of iterations.

Next we prove that (10) holds by induction. For system (1), let W_i and U_i denote the exact set Y and Z by the end of the ith outer loop, and R_i^j be X_i by the end of the jth inner loop according to Algorithm 1. We use W_i^δ, U_i^δ and R_i^δ to denote the corresponding sets for the perturbed system (7). Then $W_0 = W_0^\delta = Y_0 = \emptyset$, $U_1 = U_1^\delta = Z_1 = \emptyset$ and $R_1^0 = R_1^{\delta,0} = X_1^0 = G$. In the inner loop, for all i, j:

$$\begin{aligned} R_i^{j+1} &= U_i \cup (R_i^j \cap \mathrm{Pre}(R_1^j)), \\ R_i^{\delta,j+1} &= U_i^\delta \cup (R_i^{\delta,j} \cap \mathrm{Pre}(R_i^{\delta,j} \ominus \mathcal{B}_\delta)). \end{aligned}$$

By Proposition 3, if $\rho\varepsilon \leq \delta$, then, for $i = 1$,

$$R_1^{\delta,1} \subseteq X_1^0 \cap \mathrm{Pre}(X_1^0 \ominus \mathcal{B}_{\rho\varepsilon}) \subseteq X_1^1 \subseteq R_1^1.$$

Hence by induction, we have $R_1^{\delta,j+1} \subseteq X_1^{j+1} \subseteq R_1^{j+1}$ for all j. It follows that $\bigcup_{j\in\mathbb{Z}_{\geq 0}} R_1^{\delta,j} \subseteq \bigcup_{j\in\mathbb{Z}_{\geq 0}} X_1^j \subseteq \bigcup_{j\in\mathbb{Z}_{\geq 0}} R_1^j$, which implies $W_1^\delta \subseteq Y_1 \subseteq W_1$.

In Algorithm 3, $Z_{i+1} = Y_i \cup \underline{Z}_{i+1}$, $Y_{i+1} = Z_{i+1} \cup V_{i+1}$, and $G_{1,i+1} = \Delta Z_{i+1} \cup Z_{c,i+1}$. Hence $Y_{i+1} \cup G_{1,i+1} \cup G_{2,i+1} = (Z_{i+1} \cup V_{i+1}) \cup G_{2,i+1} \cup (\Delta Z_{i+1} \cup Z_{c,i+1}) = (Y_i \cup \underline{Z}_{i+1} \cup \Delta Z_{i+1} \cup Z_{c,i+1}) \cup (V_{i+1} \cup G_{2,i+1}) = Y_i \cup G_{1,i} \cup G_{2,i}$. It implies that $Y_i \cup G_{1,i} \cup G_{2,i} = Y_0 \cup G_{1,0} \cup G_{2,0} = [\![a_s]\!]$ for all i. For the inner loop, by Proposition 3, $G_{2,i} \cap \mathrm{Pre}(Y_{i+1} \ominus \mathcal{B}_{\rho\varepsilon}) \subseteq V_{i+1}$. Since $G_{2,i+1} = G_{2,i} \setminus V_{i+1}$, $G_{2,i+1} \cap \mathrm{Pre}(Y_{i+1} \ominus \mathcal{B}_{\rho\varepsilon}) = \emptyset$, and thus $(Y_{i+1} \cup G_{1,i+1}) \cap \mathrm{Pre}(Y_{i+1} \ominus \mathcal{B}_{\rho\varepsilon}) = [\![a_s]\!] \cap \mathrm{Pre}(Y_{i+1} \ominus \mathcal{B}_{\rho\varepsilon})$.

We also claim that $Y_i \subseteq \mathrm{Pre}(Y_i)$ for all i. For $i = 1$, $Y_1 = V_1 \subseteq G \cap \mathrm{Pre}(V_1) \subseteq \mathrm{Pre}(Y_1)$. It is clear that $\{Y_i\}$ is increasing. Suppose that $Y_i \subseteq \mathrm{Pre}(Y_i)$. Then $Y_i \subseteq \mathrm{Pre}(Y_{i+1})$, $\underline{Z}_{i+1} \subseteq \mathrm{Pre}(Y_i) \subseteq \mathrm{Pre}(Y_{i+1})$ and $V_{i+1} \subseteq \mathrm{Pre}(Y_{i+1})$. It follows that $Y_{i+1} = Y_i \cup \underline{Z}_{i+1} \cup V_{i+1} \subseteq \mathrm{Pre}(Y_{i+1})$.

By Algorithm 3, we have

$$\begin{aligned} Z_{i+1} &\subseteq Y_i \cup (G_{1,i} \cap \mathrm{Pre}(Y_i)) \\ &= (Y_i \cup G_{1,i}) \cap (Y_i \cup \mathrm{Pre}(Y_i)) \\ &\subseteq [\![a_s]\!] \cap \mathrm{Pre}(Y_i), \\ Z_{i+1} &\supseteq Y_i \cup (G_{1,i} \cap \mathrm{Pre}(Y_i \ominus \mathcal{B}_{\rho\varepsilon})) \\ &\supseteq (Y_i \cup G_{1,i}) \cap \mathrm{Pre}(Y_i \ominus \mathcal{B}_{\rho\varepsilon}) \\ &= [\![a_s]\!] \cap \mathrm{Pre}(Y_i \ominus \mathcal{B}_{\rho\varepsilon}). \end{aligned}$$

By Algorithm 1, $U_i = [\![a_s]\!] \cap \mathrm{Pre}(R_i)$ and $U_i^\delta = [\![a_s]\!] \cap \mathrm{Pre}(R_i^\delta \ominus \mathcal{B}_\delta)$. Assume that $W_i^\delta \subseteq Y_i \subseteq W_i$ for all $i \geq 1$. Then $U_{i+1}^\delta \subseteq Z_{i+1} \subseteq U_{i+1}$. Consider the inner loop of the ith outer iteration. The initial sets satisfy that $R_{i+1}^{\delta,0} \subseteq X_{i+1}^0 \subseteq R_{i+1}^0$ since $R_{i+1}^0 = U_{i+1} \cup G$, $R_{i+1}^{\delta,0} = U_{i+1}^\delta \cup G$ and $X_{i+1}^0 = Z_{i+1} \cup G$. As has been proved for the first iteration, $\bigcup_{j\in\mathbb{Z}_{\geq 0}} R_{i+1}^{\delta,j} \subseteq \bigcup_{j\in\mathbb{Z}_{\geq 0}} X_{i+1}^j \subseteq \bigcup_{j\in\mathbb{Z}_{\geq 0}} R_{i+1}^j$. Hence, $W_{i+1}^\delta \subseteq Y_{i+1} \subseteq W_{i+1}$. This proves that $\mathrm{Win}^\delta(\varphi) \subseteq Y^\varepsilon \subseteq \mathrm{Win}(\varphi)$. ∎

Theorem 1 provides a criterion for choosing the precision control parameter if the allowed bound of perturbation δ and the Lipschitz constant ρ are known a priori. In practice, one only needs to choose a sufficiently small ε according to the available computational resources. On the other hand, by starting with a large ε and iteratively reducing it until the algorithm achieves a nonempty result, Algorithm 3 can also estimate the allowable bound of the perturbations for system (1) without breaking the realizability of the given specification.

Remark 1: It is worth noting that the precision control parameter ε in the inner and outer loops of Algorithm 3 can be set to different values. Such a variant is especially useful to handle the situations that the regions labeled by a_s and a_g have tremendous difference in terms of volume. For stabilization objectives, the target region around the setpoint is commonly a minuscule portion of the state space of interest. A uniform precision will induce an overfine disretization of the entire state space and thus increase the computational complexity. Furthermore, the precision control parameters are not necessarily fixed throughout the computation, but can change with respect to the winning set obtained at each iteration. By setting a minimum threshold $\varepsilon_{\min}$ for these parameters, Theorem 1 still holds with $\varepsilon = \varepsilon_{\min}$. An

inverted pendulum control example is included in Section V to demonstrate the above discussion.

C. Extraction of switching strategies

Definition 5: Given a set $\Omega \subseteq \mathbb{R}^n$, a finite collection of sets $\mathcal{P} = \{P_1, P_2, \cdots, P_N\}$ is said to be a *partition* of Ω if i) $P_i \subseteq \Omega$; ii) $\mathrm{int}(P_i) \cap \mathrm{int}(P_j) = \varnothing$; iii) $\Omega \subseteq \bigcup_{i=1}^{N} P_i$, for all $i \in \{1, \cdots, N\}$. Each element P_i of the partition $\mathcal{P}$ is called a *cell*.

A uniform grid covering a subset $\Omega \subseteq \mathbb{R}^n$ is a partition of Ω by Definition 5. A list of interval vectors in $\mathbb{IR}^n$ forming a non-uniform grid of Ω can also be considered as a partition. By Definition 5, the interval part of the output set K of Algorithm 3 forms a partition of the system state space.

In addition to an approximation of the winning set, Algorithm 3 also records the information of the switching modes that realizes the specification φ in the form of (2). We show in the following proposition that a partition-based switching strategy of φ for system (1) can be extracted.

Proposition 4: Suppose that the assumptions in Theorem 1 hold and K is a non-empty output of Algorithm 3. Let the list of intervals $\mathcal{Y} = \{Y_1, Y_2, \cdots, Y_N\}$ be the interval part of K and $\mathcal{C} = \{C_1, C_2, \cdots, C_N\}$ be the corresponding list of valid switching modes. Then the switching strategy

$$\kappa(x) = \bigcup_{i \in N} \psi_{Y_i}(x), \tag{11}$$

where $x \in \mathbb{R}^n$, and for $i = 1, 2, \ldots, N$,

$$\psi_{Y_i}(x) = \begin{cases} \varnothing & \text{if } x \notin Y_i, \\ C_i & \text{if } x \in Y_i, \end{cases}$$

realizes the reach-and-stay specification for system (1).

Proof: Let $G := [\![a_s]\!] \cap [\![a_g]\!]$. From Algorithm 3, set K is composed of K_z and K_v. For any $([x], p) \in K_v$, under the mode p, the interval $[x]$ is either controlled inside the interval part of K_v, which are contained in G or to the set that can reach G eventually. For any $([x], p) \in K_z$, the interval $[x]$ will reach the set that can be controlled to G under the mode p. Therefore, the switching strategy (11) realizes the given specification. ∎

A switching strategy in the form of (11) is defined on a partition of the system state space, which resemble the ones generated by abstraction-based methods. Abstraction-based methods without refinement schemes mostly rely on uniform partitions. The proposed method, by contrast, adaptively partitions the state space with respect to both system dynamics and the given specification. Therefore, Algorithm 3 not only induces fewer grid points, but also returns a more accurate approximation of the real winning set than abstraction-based methods with a uniform grid size equal to ε. The comparison of the winning sets approximated by these two methods can be found in [17]. Besides, construction of abstractions and control synthesis are separated in abstraction-based methods. Whether an abstraction needs to be refined is determined by the control synthesis results. A refinement scheme on top of these two stages usually incurs repeated computation in each stage [15] without termination guarantee. In this aspect, the proposed approach provides an integrated direct control synthesis procedure, which is guaranteed to terminate and preserves robust realizability. To apply our control synthesis algorithm, only the parameter ε of Algorithm 3 has to be set to control the size of partitions, which is relatively simple.

V. AN EXAMPLE: INVERTED PENDULUM

Consider an inverted pendulum on a cart modeled by the continuous-time ODEs:

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \frac{mgl}{J_t} \sin x_1 - \frac{b}{J_t} x_2 + \frac{l}{J_t} \cos x_1 u, \end{cases} \tag{12}$$

where $x_1 = \theta$ (rad) is the angle of the pendulum to the upper vertical line, x_2 is the angle change rate $\dot{\theta}$ (rad/s), and u is the force applied to the cart; $J_t = J + ml^2$, $m = 0.2$kg, $g = 9.8\mathrm{m/s}^2$, $l = 0.3$m, $J = 0.006\mathrm{kgm}^2$, $b = 0.1\mathrm{N/m/s}$.

We aim to control the pendulum to the upright position. This specification can be written as the LTL formula $\varphi = \Box a_s \wedge \Diamond\Box a_g$, where $[\![a_g]\!] := [-0.05, 0.05] \times [-0.01, 0.01]$ and $[\![a_s]\!] := [-2, 2] \times [-3.2, 3.2]$. The sample-and-hold system of (12) with the sampling time $\tau_s = 0.01$s is used, and the control input u is chosen from the finite set $\mathcal{M} = 0.05\{-10, -9, \cdots, 9, 10\}$. Then (12) can be treated as a discrete-time switched system with finite modes.

A local growth bound[1] is used to construct the inclusion function required to apply the proposed algorithm:

$$\beta(\eta, u) = e^{L(u)\tau_s}\eta,\ L(u) = \begin{bmatrix} 0 & 1 \\ \sqrt{24.5^2 + 12.5^2 u^2} & -4.17 \end{bmatrix},$$

where $\eta = [\eta_1, \eta_2]$ is the grid width.

In this case, the target stabilization area G is tiny compared to the entire state space X. In order to maintain the pendulum angle and angle change rate in the region G, the value of the precision control parameter ε has to be determined according to the size of G. Thus, we use a precision $\varepsilon = 0.001$ for the CPRED of the inner loop. Since the state space X is nearly 40 times the size of G, the partition of X will contain a huge number of cells if a uniform precision $\varepsilon = 0.001$ is used in Algorithm 3. To obtain an acceptable computational complexity, we use a relative precision, which is determined with respect to the size of the winning set throughout iterations, for the outer-loop CPRED. The inner loop precision reflects the bound of the perturbation that can be tolerated by the resulting switching strategy.

We use ROCS [24] to solve this control problem with these parameter settings. For an initial condition $(1, 1)$, the closed-loop simulation result (see Figure 1) shows that, applying the extracted switching strategy, the angle of pendulum is stabilized to zero with a steady-state error of 0.05.

This system is neither globally asymptotically stable nor incrementally asymptotically stable around the upright position. Hence, abstraction-based methods using bi-simulation relations [11] do not apply while over-approximations [13], [12], [25], [23] based on uniform grids can be used. To achieve equivalent stabilization precisions, the grid size

[1] See [23] for the definition and the construction.

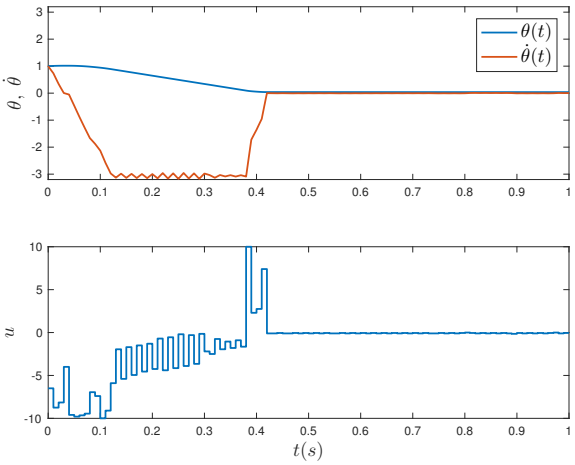

Fig. 1: Closed-loop simulation with the initial condition $(\theta_0, \dot{\theta}_0) = (1, 1)$ for system (12).

needs to be at least 0.001 according to the size of the stabilization area. Using uniform partitions as in abstraction-based methods, the entire safe region $[\![a_s]\!]$ is discretized to overall 2.56×10^7 cells with grid points of width 0.001. Using SCOTS [26], computation of the abstraction lasts for more than 12 hours without returning any result. In contrast, our algorithm generates a winning set covering most of the state space in around 480 seconds with 26340 partitions.

VI. CONCLUSIONS

This paper presented a synthesis algorithm for discrete-time switched systems with reach-and-stay objectives. The proposed algorithm works under a fixed-point iteration scheme developed for finite-state game graphs [21], but it is performed directly over continuous state spaces of switched systems. We proved that such a fixed-point iteration is sound and complete even for infinite-state systems. As the algorithm proceeds, the continuous state space of a given switched system is adaptively partitioned into a finite number of interval vectors with respect to both specifications and system dynamics, which is achieved by incorporating the interval branch-and-bound scheme. With valid switching modes recorded during partitioning, a memoryless controller can be extracted immediately after the algorithm terminates. This is in contrast with abstraction-based methods in which control synthesis is performed indirectly on a finite abstraction of the original system. As a major advantage over abstraction-based methods, the proposed algorithm is robustly complete in the sense that it returns a switching strategy for any robustly realizable specifications. Moreover, our algorithm demonstrates higher efficiency than abstraction-based methods as the size of the partition is reduced considerably, especially when the target region has to be controlled invariant with high precision as in most of the stablization objectives.

Future work will concentrate on solving more general LTL control synthesis problems under the proposed direct synthesis framework. More general dynamics such as continuous-time and general nonlinear systems will also be studied.

ACKNOWLEDGMENT

The research was supported in part by the Natural Sciences and Engineering Council (NSERC) of Canada and the Canada Research Chairs (CRC) Program.

REFERENCES

[1] D. Bertsekas and I. Rhodes, "On the minimax reachability of target sets and target tubes," *Automatica*, vol. 7, no. 2, pp. 233–247, 1971.
[2] D. P. Bertsekas, "Infinite-time reachability of state-space regions by using feedback control," *IEEE Trans. Automat. Contr.*, vol. 17, no. 5, pp. 604–613, 1972.
[3] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.
[4] S. Rakovic, E. Kerrigan, D. Mayne, and J. Lygeros, "Reachability analysis of discrete-time systems with disturbances," *IEEE Trans. Automat. Contr.*, vol. 51, no. 4, pp. 546–561, 2006.
[5] L. Fribourg and R. Soulat, *Control of Switching Systems by Invariance Analysis: Application to Power Electronics*. Wiley-ISTE, 2013.
[6] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1370–1381, 2009.
[7] E. Frazzoli, M. A. Dahleh, E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1077–1091, 2005.
[8] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.
[9] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT press, 2008.
[10] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Proc. of HSCC*, 2003, pp. 498–513.
[11] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE Trans. Automat. Contr.*, vol. 55, no. 1, pp. 116–126, 2010.
[12] J. Liu, N. Ozay, U. Topcu, and R. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Trans. Automat. Contr.*, vol. 58, no. 7, pp. 1771–1785, 2013.
[13] M. Zamani, G. Pola, M. M. Jr., and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Trans. Automat. Contr.*, vol. 57, no. 7, pp. 1804–1809, 2012.
[14] J. Liu, "Robust abstractions for control synthesis: Robustness equals realizability for linear-time properties," in *Proc. of HSCC*, 2017.
[15] P. Nilsson, N. Ozay, and J. Liu, "Augmented finite transition systems as abstractions for control synthesis," *Discret. Event Dyn. Syst.*, vol. 27, no. 2, pp. 301–340, 2017.
[16] L. Jaulin, *Applied Interval Analysis*. Springer Science & Business Media, 2001.
[17] Y. Li and J. Liu, "Invariance control synthesis for switched nonlinear systems: An interval analysis approach," *IEEE Trans. Automat. Contr.*, pp. 1–1, 2017.
[18] S. Gao, S. Kong, W. Chen, and E. M. Clarke, "Delta-complete analysis for bounded reachability of hybrid systems," *CoRR*, vol. abs/1404.7171, 2014.
[19] F. Blanchini, "Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions," *IEEE Trans. Automat. Contr.*, vol. 39, no. 2, pp. 428–433, 1994.
[20] A. Tarski, "A lattice-theoretical fixpoint theorem and its applications," *Pacific J. Math.*, vol. 5, no. 2, pp. 285–309, 1955.
[21] L. de Alfaro, T. Henzinger, and R. Majumdar, "From verification to control: dynamic programs for omega-regular objectives," in *Proc. of Annu. IEEE Symp. Log. Comput. Sci.*, 2001, pp. 279–290.
[22] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Springer, 2009.
[23] G. Reissig, A. Weber, and M. Rungger, "Feedback refinement relations for the synthesis of symbolic controllers," *IEEE Trans. Automat. Contr.*, vol. 62, no. 4, pp. 1781 – 1796, 2017.
[24] Y. Li and J. Liu, "ROCS: A robustly complete control synthesis tool for nonlinear dynamical systems," in *Proc. of Hybrid Systems: Computation and Control (HSCC)*, to appear.
[25] Y. Li, J. Liu, and N. Ozay, "Computing finite abstractions with robustness margin via local reachable set over-approximation," in *Proc. of IFAC Conf. Analysis and Design of Hybrid Systems (ADHS)*, 2015.
[26] M. Rungger and M. Zamani, "SCOTS: a tool for the synthesis of symbolic controllers," in *Proc. of HSCC*, 2016, pp. 99–104.