Synthesis of Reactive Switching Protocols From Temporal Logic Specifications

Jun Liu, *Member, IEEE*, Necmiye Ozay, *Member, IEEE*, Ufuk Topcu, *Member, IEEE*, and Richard M. Murray, *Fellow, IEEE*

***Abstract*—We propose formal means for synthesizing switching protocols that determine the sequence in which the modes of a switched system are activated to satisfy certain high-level specifications in linear temporal logic (LTL). The synthesized protocols are robust against exogenous disturbances on the continuous dynamics and can react to possibly adversarial events (both external and internal). Finite-state approximations that abstract the behavior of the underlying continuous dynamics are defined using finite transition systems. Such approximations allow us to transform the continuous switching synthesis problem into a discrete synthesis problem in the form of a two-player game between the system and the environment, where the winning conditions represent the high-level temporal logic specifications. Restricting to an expressive subclass of LTL formulas, these temporal logic games are amenable to solutions with polynomial-time complexity. By construction, existence of a discrete switching strategy for the discrete synthesis problem guarantees the existence of a switching protocol that can be implemented at the continuous level to ensure the correctness of the nonlinear switched system and to react to the environment at run time.**

***Index Terms*—Formal synthesis, hybrid systems, linear temporal logic, switching protocols, temporal logic games.**

I. Introduction

THE objective of this paper is to synthesize switching protocols that determine the sequence in which the modes of a switched system are activated to satisfy certain high-level specifications formally stated in linear temporal logic (LTL). Different modes may correspond to, for example, the evolution of the system under different, pre-designed feedback controllers [24], so-called motion primitives in robot motion planning [12], or different configurations of a system (e.g., different gears in a car or aerodynamically different phases of a flight). Each of these modes may meet certain specifications but not necessarily the complete, mission-level specification the system needs to satisfy. The purpose of the switching protocol is to identify a switching sequence such that the resulting switched system satisfies the mission-level specification.

We are interested in designing *open* systems that can sense their environment and interact with it by appropriately (as encoded by the specification) reacting to sensory inputs. Given a family of system models, in the form of ordinary differential equations potentially with bounded exogenous disturbances, a description of the sensed environment, and an LTL specification, our approach builds on a hierarchical representation of the system in each mode and the environment. In particular, we model environment as transition systems, which can be constructed through observations of possibly continuous signals. While the continuous evolution of the system is accounted for at the low level, the higher level is composed of a finite-state approximation of the continuous evolution. The switching protocols are synthesized using the high-level, discrete models. Simulation-type relations [1] between the continuous and discrete models guarantee that the correctness of the synthesized switching protocols is preserved in the continuous implementation.

Our approach relies on a type of finite-state approximation for continuous nonlinear systems, namely over-approximation. Roughly speaking, we call a finite transition system $\mathcal{T}$ an over-approximation of the continuous system if for each transition in $\mathcal{T}$, there is a possibility for continuously implementing the strategy, in spite of either the exogenous disturbances or the coarseness of the approximation. We account for the mismatch between the continuous model and its over-approximation as nondeterminism and treat it as adversary. Consequently, the corresponding switching protocol synthesis problem is formulated as a two-player temporal logic game (see [33] and references therein and the pioneering work in [9]). This game formulation allows us to incorporate environment signals that do not affect the dynamics of the system but constrain its behavior through the specification. Within the game, environment is also treated as adversarial, that is, we aim to synthesize controllers that guarantee the satisfaction of the specification even against the worst-case behavior of the environment.

The main contributions of the current paper are in (1) proposing a framework for switching protocol synthesis for continuous-time nonlinear switched systems, potentially subject to exogenous disturbances, from LTL specifications, and (2) synthesis of controllers that can continuously sense their environment and react to the changes in the environment according to the specification. In contrast to much of prior work

Manuscript received September 16, 2011; revised August 31, 2012, December 31, 2012; accepted January 01, 2013. Date of publication February 07, 2013; date of current version June 19, 2013. This work was supported in part by the NSERC of Canada, the FCRP consortium through the Multiscale Systems Center (MuSyC), AFOSR Award FA9550-12-1-031, and the Boeing Corporation. Recommended by Associate Editor D. E. Dullerud.

J. Liu is with the Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield S1 3JD, U.K. (e-mail: j.liu@shef.ac.uk).

N. Ozay and R. M. Murray are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: necmiye@caltech.edu; murray@cds.caltech.edu).

U. Topcu is with the Department of Electrical and Systems Engineering, the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: utopcu@seas.upenn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2013.2246095

(described in more detail below), we consider open systems that can maintain ongoing interaction with some external environment and synthesize controllers that are reactive to this environment. The use of LTL enables us to handle a wide variety of specifications beyond mere safety and reachability. The game formulation allows us to account for a potentially adversarial, *a priori* unknown environment in which the system operates (and therefore its correctness needs to be interpreted with respect to the allowable environment behaviors). While the general game structure considered in this paper can handle full LTL specifications, such games with complete LTL specifications are known to have high computational complexity [35]. Therefore, we focus on an expressive fragment of LTL, namely Generalized Reactivity (1) (GR(1)) specifications, for which there exist synthesis algorithms with favorable computational complexity [33].

Fragments of the switching protocol synthesis problem considered here have attracted considerable attention. We now give a very brief overview of some of the existing work as it ties to the proposed methodology (a thorough survey is beyond the scope of this paper). Stability problems for switched systems arise naturally in the context of switching control and considerable work has been focused on designing switching controllers for stability. Hespanha and Morse [16] propose the notion of average dwell time, generalizing that of dwell time considered by Morse [29]. Both dwell time and average dwell time are design parameters that can be chosen to achieve stability when switching between stabilizing controllers. Jha *et al.* [18] focuses on switching logics that guarantee the satisfaction of certain safety and dwell-time requirements. Taly and Tiwari [39], Cámara *et al.* [8], Asarin *et al.* [2], and Koo *et al.* [21] consider either switching controller synthesis for safety [8], reachability [21], or a combination of safety and reachability properties [2], [39]. Joint synthesis of switching logics and feedback controllers for stability are studied by Lee and Dullerud [23]. The work by Frazzoli *et al.* [12] on the concatenation of a number of motion primitives from a finite library to satisfy certain reachability properties constitutes an instance of switching protocol synthesis problem. Our work also has strong connections with the automata-based composition of the so-called interfaces that describe the functionality and the constraints on the correct behavior of a system [42]. Also related is the work by Moor and Davoren [28] who use modal logic to design switching control that is robust to uncertainty in the differential equations. The objective there is to determine a control sequence to switch among the family of differential equations to satisfy a specification that includes safety and event sequencing. Our work extends these results both in terms of the family of models that can be handled and the expressivity of the specification language used.

More broadly, this paper can be seen in the context of abstraction-based, hierarchical approaches to controller synthesis for continuous and hybrid systems from high-level specifications given in terms of LTL [4], [11], [15], [20], [38]. Either by limiting the type of dynamical systems considered [4], [15], [20] or by considering a rich (e.g., unconstrained) control input [11], [38], which can arbitrarily steer the system, these papers obtain deterministic abstractions for the underlying systems and use model-checking based methods for synthesis. The work by Tûmová *et al.* [41] extends this framework by allowing piecewise affine systems and nondeterministic transitions in the abstraction. While allowing more flexibility in constructing the abstraction, the presence of nondeterministic transitions render standard model checking tools no longer applicable for finding a control strategy. Kloetzer and Belta [19] investigate this and propose a solution inspired from (infinite) two-player LTL games. Compared to these results, the control authority in our framework is fairly limited (i.e., only control input is the mode of the switched system) which leads to a more challenging problem.

Beyond combining ideas from and extending results in switching protocol synthesis and abstraction-based controller synthesis, the main difference between the aforementioned papers [2], [4], [8], [11], [12], [15], [16], [18]–[21], [23], [28], [29], [38], [39], [41], [42] and the work presented here is that, in those papers, the control strategy is *non-reactive* in the sense that its behavior does not take into account external events from the environment at *run time*. Such runtime reactiveness is crucial for satisfying safety properties. For instance, in cases of emergency (e.g., in traffic when a pedestrian jumps on the street) the system might need to take immediate action, or when a failure occurs, the system might need to deactivate the faulty parts promptly for safe operation. Reactive controller synthesis, in particular by using GR(1) games, has been considered before by Kress-Gazit *et al.* [22] and Wongpiromsarn *et al.* [44]. Kress-Gazit *et al.* [22] consider systems with fully actuated dynamics ($\dot{x} = u$), for which *bisimilar* continuous implementations of discrete plans are guaranteed to exist. Although the environment is continuously sensed in [22], one limitation of the method is that the controller is allowed to take actions against the environment only when certain locative propositions are satisfied, hence the system can not synchronously react to the environment at run time. Wongpiromsarn *et al.* [44] focus on discrete-time linear time-invariant models with exogenous disturbances and propose a receding horizon framework for alleviating the computational complexity. Reasoning of continuous trajectories and environment behavior in continuous time is not a concern in [44] either, as discrete-time models are considered. Abstractions considered in both papers are deterministic in the sense that every transition between discrete states can be implemented, whereas in the current paper we focus on over-approximations, which are potentially easier to compute for switched nonlinear systems, where the controls are limited to the discrete modes.

Contrary to existing work, one of the major contributions of the paper is to achieve runtime reactiveness of the continuous-time switching controller. To this end, we explicitly model the environment and take into account possibly asynchronous interactions between the system dynamics and the environment. This is accomplished by defining product transition systems from the abstractions and augmenting these transition systems with additional liveness conditions to enforce progress in accordance with the underlying dynamics.

II. Preliminaries

In this section, we introduce linear temporal logic (LTL) [27], [34] as the specification language to formally specify system properties.

A. LTL Syntax and Semantics

Standard LTL is built upon a finite set of atomic propositions, logical operators $\neg$ (negation) and $\vee$ (disjunction), and the temporal modal operators $\bigcirc$ (next) and $\mathcal{U}$ (until). LTL is a rich specification language that can express many desired properties, including safety, reachability, invariance, response, and/or a combination of these [27].

Syntax of LTL: Formally, the set of LTL formulas over a finite set of atomic propositions Π can be defined inductively as follows:

(1) any atomic proposition $\pi \in \Pi$ is an LTL formula;
(2) if φ and ψ are LTL formulas, so are $\neg\varphi$, $\bigcirc\varphi$, $\varphi \vee \psi$, and $\varphi \mathcal{U} \psi$.

Additional logical operators, such as $\wedge$ (conjunction), $\rightarrow$ (material implication), and temporal modal operators $\Diamond$ (eventually), and $\Box$ (always), are defined by:

(a) $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$;
(b) $\varphi \rightarrow \psi := \neg\varphi \wedge \psi$;
(c) $\texttt{True} := p \vee \neg p$, where $p \in \Pi$;
(d) $\Diamond\varphi := \texttt{True}\mathcal{U}\varphi$;
(e) $\Box\varphi := \neg\Diamond\neg\varphi$.

A *propositional formula* is one that does not include any temporal operators.

Semantics of LTL: An LTL formula over Π is interpreted over *ω-words*, i.e. infinite sequences, in 2^{Π}. Let $w = w_0 w_1 w_2 \cdots$ be such a word. The satisfaction of an LTL formula φ by w at position i, written $w_i \models \varphi$, is defined recursively as follows:

(1) for any atomic proposition $\pi \in \Pi$, $w_i \models \pi$ if and only if $\pi \in w_i$;
(2) $w_i \models \neg\varphi$ if and only if $w_i \not\models \varphi$;
(3) $w_i \models \bigcirc\varphi$ if and only if $w_i \models \varphi$;
(4) $w_i \models \varphi \vee \psi$ if and only if $w_i \models \varphi$ or $w_i \models \psi$;
(5) $w_i \models \varphi \mathcal{U} \psi$ if and only if there exists $j \geq i$ such that $w_j \models \psi$ and $w_k \models \varphi$ for all $k \in [i, j)$.

The word w is said to satisfy φ, written $w \models \varphi$, if $w_0 \models \varphi$.

LTL Without Next Step: We denote the subset of the logic LTL without the next operator by $\mathrm{LTL}_{\setminus\bigcirc}$.

B. Finite Transition System

Definition 1: A *finite transition system* is a tuple $\mathcal{T} := (\mathcal{Q}, \mathcal{Q}_0, \rightarrow)$, where $\mathcal{Q}$ is a finite set of states, $\mathcal{Q}_0 \subseteq \mathcal{Q}$ is a set of initial states, and $\rightarrow \subseteq \mathcal{Q} \times \mathcal{Q}$ is a transition relation. Given states $q, q' \in \mathcal{S}$, we write $q \rightarrow q'$ if the transition relation $\rightarrow$ contains the pair (q, q').

We assume that for every state $q \in \mathcal{Q}$, there exists a state $q' \in \mathcal{Q}$ such that $q \rightarrow q'$. Let $h_Q : \mathcal{Q} \rightarrow 2^{\Pi}$ be an *observation* map which maps the state space $\mathcal{Q}$ to a finite set of propositions.

Definition 2: An *execution* $\rho : \mathbb{Z}^+ \rightarrow \mathcal{Q}$ of a finite transition system $\mathcal{T} = (\mathcal{Q}, \mathcal{Q}_0, \rightarrow)$ is an infinite sequence $\rho = q_0 q_1 q_2 \cdots$, where $\rho(0) = q_0 \in \mathcal{Q}_0$ and $\rho(i) = q_i \in \mathcal{Q}$ and $\rho(i) \rightarrow \rho(i+1)$ $(i \in \mathbb{Z}^+)$. The *word* produced by an execution ρ is $w_\rho = w_\rho(0) w_\rho(1) w_\rho(2) \cdots$, where $w_\rho(i) = h_Q(\rho(i))$ for all $i \geq 0$. An execution ρ is said to satisfy an LTL formula φ, written $\rho \models \varphi$, if and only if the word it produces satisfies φ. If all executions of $\mathcal{T}$ satisfy φ, we say that the finite transition system $\mathcal{T}$ satisfies φ and write $\mathcal{T} \models \varphi$.

C. Continuous-Time Signals

Let X be a set (infinite or finite) and $h_X : X \rightarrow 2^{\Pi}$ be an *observation* map which maps X to a finite set of propositions. A *continuous-time signal* $s \in X$ is a function $s : \mathbb{R}^+ \rightarrow X$.

Definition 3: We say that a signal $s \in X$ is of *finite variability* (on bounded intervals of $\mathbb{R}^+$) under observation h_X if there exists an infinite number of non-overlapping intervals $\mathcal{I} = I_1, I_2, \cdots$ such that $\cup_{i=1}^{\infty} I_i = \mathbb{R}^+$, $h_X(s(t)) = h_X(s(t'))$ for all $t, t \in I_i$, and $l(I_j) \rightarrow \infty$ as $j \rightarrow \infty$, where $l(I_j)$ denotes the left end-point of I_j.

The above definition is similar to the notion of finite variability for continuous-time Boolean signals [26]. Here the observation map is explicit and original signals need not to be of finite variability. In the rest of the paper, unless otherwise stated, we restrict our attention to signals that are of finite variability under some observation map.

Definition 4: Let s be continuous-time signal in X. The *word* produced by s is a sequence $w_s = w_s(0) w_s(1) w_s(2) \cdots$, defined recursively as follows:

- $w_s(0) = h_X(s(0))$;
- $w_s(k) = h_X(s(\tau_k^+)) = \lim_{t \rightarrow \tau_k^+} h_X(s(t))$ for all $k \geq 1$ such that $\tau_k < \infty$, where τ_k is defined by $\tau_0 = 0$ and $\tau_k = \inf\{t : t > \tau_{k-1}, h_X(s(t)) \neq w_s(k-1)\}$ for all $k \geq 1$; and
- $w_s(k) = w_s(k-1)$ for all k such that $\tau_k = \infty$.

The word w_s is said to be *well-defined* if $\tau_k \rightarrow \infty$ as $k \rightarrow \infty$. The signal s is said to satisfy an LTL formula φ, written $\rho \models \varphi$ $(s \models \varphi)$, if and only if the word it produces is well-defined and satisfies φ.

Intuitively, the word generated by a signal is exactly the sequence of sets of propositions satisfied by the signal as time evolves. This definition is consistent with that of [20, Definition 4]. In the above definition, it is assumed that the limit $h_X(s(\tau^+)) = \lim_{t \rightarrow \tau^+} h_X(s(t))$ for all $\tau \geq 0$. This would exclude certain unrealistic signals such as those with observations 1 on rationals and 0 on irrationals out of the scope. It is easy to check that the above definition is well-posed and gives well-defined words for signals of finite variability.

III. Problem Formulation

In this section, we introduce the system models and formulate the main problem studied in this paper.

A. Continuous-Time Switched Systems

Consider a family of nonlinear systems,

$$\dot{x}(t) = f_p\left(x(t), d(t)\right), \quad p \in \mathcal{P}, \quad t \geq 0, \tag{1}$$

where $x(t) \in X \subseteq \mathbb{R}^n$ is the state at time t and $d(t) \in D \subseteq \mathbb{R}^d$ is an exogenous disturbance, $\mathcal{P}$ is a finite index set, and $\{f_p : p \in \mathcal{P}\}$ is a family of nonlinear vector fields satisfying the usual conditions to guarantee the existence and uniqueness of

solutions for each of the subsystems in (1). A *switched system* generated by the family (1) can be written as

$$\dot{x}(t) = f_{\sigma(t)}\left(x(t), d(t)\right), \quad t \geq 0, \tag{2}$$

where σ is a switching signal taking values in $\mathcal{P}$. The value of σ at a given time t may depend on t or $x(t)$, or both, or may be generated by using more sophisticated design techniques [24]. Given sets of initial states $X_0 \subseteq X$ and initial modes $\mathcal{P}_0 \subseteq \mathcal{P}$, *solutions* to (2) are pairs (x, σ) that satisfy (2) for all $t \geq 0$ and $(x(0), \sigma(0)) \in X_0 \times \mathcal{P}_0$.

In the above formulation, σ is the only controllable variable. This captures different situations where control inputs are limited to a finite number of quantized levels, e.g., $\{u_p : p \in \mathcal{P}\} \subseteq \mathbb{R}^m$, or chosen from a family of feedback controller $\{u_p(t) = K_p(x(t)) : p \in \mathcal{P}\}$. In addition, depending on different applications, each mode in (1) may represent, for example, a control component [42], a motion primitive (which belongs to, e.g., a finite library of flight maneuvers [12], or a set of pre-designed behaviors [40]), or, in general, an operating mode of a multi-modal dynamical system[18], [39]. To achieve complex tasks, it is often necessary to compose these basic components.

B. Environment

In this paper, we use *environment* to refer to the factors that are relevant to the operation of the system, but do not impact its dynamics directly, i.e., not explicitly present in (1) or (2). Such factors are not necessarily controlled by the system, e.g., obstacles, traffic lights, weather conditions, software and hardware faults and failures. As such, they are usually treated as adversaries.

More specifically, the environment consists a set of environment variables, compactly denoted as $\mathbf{e}$, which can take values in a set Z. While Z is not necessarily a finite set, we assume that there exists an observation map $h_Z : Z \to \mathcal{E}$ which maps states in Z to a finite set $\mathcal{E}$ of observations. Real-time properties of the environment can be captured by the definition of environment signals. An *environment signal* is a function $\zeta : \mathbb{R}^+ \to Z$.

We restrict our attention to environment signals that are of finite variability (in the sense of Definition 3) under the observation h_Z. For such signals, we introduce the following notion to capture their behavior as a whole.

Definition 5: The *environment transition system* is a tuple $\mathcal{T}_e := (\mathcal{E}, \mathcal{E}_0, \to)$, where $\mathcal{E}$ is a finite set of states, $\mathcal{E}_0 \subseteq \mathcal{E}$ is the set of initial states, $\to \subseteq \mathcal{E} \times \mathcal{E}$ is a transition relation defined by $e \to e'$ if and only $e \neq e'$ and there exists an environment signal ζ and some $\tau > 0$ such that $h_Z(\zeta(\tau)) = e'$ and $\lim_{t \to \tau^-} h_Z(\zeta(t)) = e$.

In other words, the environment transition system $\mathcal{T}_e$ captures all possible changes in the environment (described by a set of environment signals). Note that this does not mean that we have complete knowledge of the environment. Rather, $\mathcal{T}_e$ constitutes a finite representation of the environment model. As will be shown in the next section, our knowledge about the allowable environment behavior will be reflected in the specification as an environment assumption.

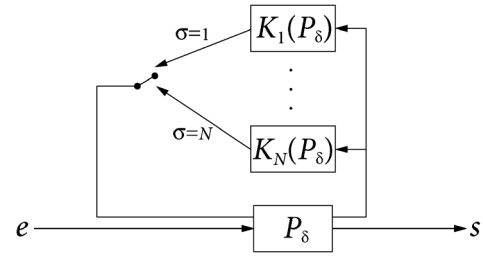


Fig. 1. P_δ represents a plant subject to exogenous disturbances, $\{K_i(P_\delta) : i = 1, \cdots, N\}$ is a family of controllers, e represents the environment, which does not directly impact the dynamics of the system but constrains its behavior through the specification. The objective is to design σ such that the overall system satisfies a high-level specification φ expressed in LTL.

C. Problem Formulation

The goal of this paper is to propose methods for automatically synthesizing a switching protocol σ such that solutions of the resulting switched system (2) satisfy, by construction, a given linear temporal logic (LTL) specification, for all possible exogenous disturbances. In addition, as encoded by the specification, the switching protocol σ should be able to react to possibly adversarial events (both internal and external) captured by the environment, in the sense that if a change in the environment is detected, the system can react immediately by possibly switching to a different mode. A schematic description of the problem is shown in Fig. 1.

To formally state the problem, let $\Pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$, the finite set of atomic propositions, be defined as $\Pi := \Pi_X \times \mathcal{E} \times \mathcal{P}$, where Π_X is a finite set of subsets of X. The observation map

$$h_{X \times Z \times \mathcal{P}} : X \times Z \times \mathcal{P} \to 2^{\Pi}, \tag{3}$$

is defined by $h(x, z, p) = (h_X(x), h_Z(z), p)$, where $(x, z, p) \in X \times Z \times \mathcal{P}$ and $h_X : X \to \Pi_X$ is defined by $h_X(x) = \{\pi \in \Pi_X : x \in \pi\}$, i.e., the set of elements in Π_X that contain x.

Defintion 6: A *trajectory* $s : \mathbb{R}^+ \to X \times Z \times \mathcal{P}$ of the switched system (2) and its environment is a triplet $s = (x, \zeta, \sigma)$, where (x, σ) are solutions to (2) and ζ is an environment signal.

Now we are ready to formally state the switching synthesis problem.

Continuous Switching Synthesis Problem: Given a family of continuous-time subsystems in (1), a model of the environment $\mathcal{T}_e$, and a specification φ expressible in $\mathrm{LTL}_{\setminus\bigcirc}$ of the form

$$\varphi := (\varphi_e \to \varphi_s), \tag{4}$$

where φ_e is the *environment assumption* that encodes the knowledge about the allowable environment behavior, and φ_s encodes the desired behavior of the system, synthesize a *reactive* switching protocol σ that

(i) generates only correct trajectories $s = (x, \zeta, \sigma)$ in the sense that $s \models \varphi$ for all allowable environment behavior; and

(ii) reacts to environment changes in real time in the sense that a switching decision (a change in $\sigma(t)$) can be made immediately, whenever there is a change in the environment (a change in $\zeta(t)$).

Overview of Solution Strategy: In the next few sections, we use a hierarchical approach to solve the switching synthesis problem in three steps:

(i) establish finite-state approximations which abstract the family of systems (1) in each mode;
(ii) formulate a discrete synthesis problem based on this finite abstraction;
(iii) synthesize a switching protocol by solving the discrete synthesis problem. This switching protocol, when implemented, gives a solution to the continuous switching synthesis problem.

More specifically, in Section IV, we introduce a type of discrete abstractions called *over-approximations*, which *conservatively* approximates the continuous dynamics. This abstraction, together with a finite representation of the environment and an LTL specification, defines a discrete synthesis problem.

In Section V, we reformulate and solve the discrete synthesis problem as a two-player temporal logic game. While solving two-player games with general LTL winning conditions is known to have prohibitively high computational complexity [35], we can restrict ourselves to an expressive fragment of LTL, namely Generalized Reactivity (1), to achieve favorable computational complexity [33]. Continuous implementations of the switching protocol, together with its correctness and reactiveness, are discussed in Section VI. In Section VII, we present four illustrative examples to demonstrate our results.

IV. Discrete Representation of the Synthesis Problem

In this section, we define over-approximations for the continuous dynamics (1) and formulate the discrete synthesis problem.

A. Finite-State Approximations of Dynamics

Abstractions for each of the subsystems in (1) are induced by an abstraction map $T : X \to \mathcal{Q}$, which maps each state $x \in X$ into a finite set $\mathcal{Q} := \{q_i : i = 1, \cdots, M\}$ and initial states $x_0 \in X_0$ into $\mathcal{Q}_0 \subseteq \mathcal{Q}$. The map T essentially defines a partition of the state space $\mathbb{R}^n$ by $\{T^{-1}(q) : q \in \mathcal{Q}\}$. We shall refer to elements in $\mathcal{Q}$ as discrete states of an abstraction. Since we do not distinguish the continuous states within $T^{-1}(q)$ for a given q, we require T to preserve propositions of interest in the sense that

$$T(x) = T(y) \Longrightarrow h_X(x) = h_X(y), \quad \forall x, y \in X, \qquad (5)$$

where h_X is as defined in (3). The proposition preserving abstraction map T induces an observation map $h_{\mathcal{Q}\times\mathcal{E}\times\mathcal{P}}$ on the discrete state space $\mathcal{Q} \times \mathcal{E} \times \mathcal{P}$ as

$$h_{\mathcal{Q}\times\mathcal{E}\times\mathcal{P}}(q, e, p) = (h_X(x), h_Z(z), p),$$

where $(q, e, p) \in \mathcal{Q}\times\mathcal{E}\times\mathcal{P}$ and (x, z) are such that $x \in T^{-1}(q)$ and $h_Z(z) = e$.

Now consider a family of finite transition systems

$$\mathcal{T}_q := \left\{(\mathcal{Q}, \mathcal{Q}_0, \xrightarrow{p}) : p \in \mathcal{P}\right\}, \qquad (6)$$

where $\mathcal{Q}$ is as defined above, $\mathcal{Q}_0 \subseteq \mathcal{Q}$ is a set of initial states, and $\xrightarrow{p} \subseteq \mathcal{Q} \times \mathcal{Q}$ is a transition relation.

Definition 7: The family of finite transition systems in (6) is said to be an *over-approximation* for (1) if the following two statements hold.

(i) Given states $q, q' \in \mathcal{Q}$ such that $q' \neq q$, there is a transition $q \xrightarrow{p} q'$, if there exists $x_0 \in T^{-1}(q)$, $\tau > 0$, and some exogenous disturbance $d : [0, \tau] \to D \subseteq \mathbb{R}^d$ such that the corresponding trajectory ξ of the pth subsystem of (1) starting from x_0, i.e., $\xi : [0, \tau] \to \mathbb{R}^n$ with

$$\xi(0) = x_0, \quad \dot{\xi}(t) = f_p(\xi(t), d(t)), \quad \forall t \in (0, \tau),$$

satisfies

$$\xi(\tau) \in T^{-1}(q'), \xi(t) \in T^{-1}(q) \cup T^{-1}(q'), t \in [0, \tau].$$

(ii) For any $q \in \mathcal{Q}$, there is a self-transition $q \xrightarrow{p} q$, if there exists $x_0 \in T^{-1}(q)$ and some exogenous disturbance $d : [0, \infty) \to D \subseteq \mathbb{R}^d$ such that the complete trajectory ξ of the pth subsystem of (1) on $[0, \infty)$ starting from x_0 is contained in $T^{-1}(q)$.

Intuitively, in an over-approximation, a discrete transition $q \xrightarrow{p} q'$ is included as long as there is a possibility (either induced by disturbances or a coarse partition) for the pth subsystem to implement the transition. In the above definition, time is abstracted out in the sense that we do not care how much time it takes to reach one discrete state from another.

In Definition 7, if (ii) is not satisfied, i.e., all trajectories of the pth subsystem starting from $T^{-1}(q)$ leave it eventually, we say that the region $T^{-1}(q)$ or state q is *transient* for mode p. This notion of transience can be generalized for a subset of modes in a natural way. A discrete state $q \in \mathcal{Q}$ is said to be *transient* for a subset of modes $\mathcal{P}_s \subseteq \mathcal{P}$, if all trajectories of the switched system (2) starting from $T^{-1}(q)$ leave it eventually under arbitrary switching signals taking values in $\mathcal{P}_s$. A similar notion of transience is proposed in [4] in the context of autonomous piecewise multi-affine system. Here we define it for switched system with possible exogenous disturbances and extend it to ensure progress under switching among a certain set of modes, which is particularly important when reasoning about the interactions of the system with the environment at the discrete level.

The following result uses a barrier certificate [37] type condition for verifying a discrete state being transient.

Proposition 1: Given $q \in \mathcal{Q}$ and assume that $T^{-1}(q)$ is bounded. The discrete state q is transient on a family of modes $\mathcal{P}_s \subseteq \mathcal{P}$, if there exists a $\mathcal{C}^1$ function $B : \mathbb{R}^n \to \mathbb{R}$ and $\varepsilon > 0$ such that

$$\frac{\partial B}{\partial x}(x) f_p(x, d) \leq -\varepsilon,$$
$$\forall x \in T^{-1}(q), \quad \forall p \in \mathcal{P}_s, \quad \forall d \in D. \qquad (7)$$

Proof: Assume, for the sake of contradiction, that there exists some $x_0 \in T^{-1}(q)$ and a switching signal σ_s taking values in $\mathcal{P}_s$ such that the complete trajectory ξ of (2) on $[0, \infty)$ starting from x_0 is contained in $T^{-1}(q)$. Condition ((7)) implies that

$$\frac{dB(\xi(t))}{dt} = \frac{\partial B}{\partial x}(\xi(t)) f_{\sigma_s(t)}(\xi(t)) \leq -\varepsilon, \qquad (8)$$

since $\xi(t) \in T^{-1}(q)$ and $\sigma_s(t) \in \mathcal{P}_s$ for all $t \geq 0$. Integrating this equation would imply $\lim_{t\to\infty} B(\xi(t)) = -\infty$. Yet this leads to a contradiction, as $B(\xi(t))$ should be bounded if $\xi(t)$ remains in the bounded set $T^{-1}(q)$. ■

Next, we provide pseudocode for an algorithm for computing finite-approximations of (1). We focus on a particularly useful case where $T^{-1}(q)$ are convex polytopes for all q. The main subroutines of the algorithm are as follows:

- $Neighbors(T^{-1}(q_j))$: returns neighbors of the polytope $T^{-1}(q_j)$, i.e., polytopes that share a common facet with $T^{-1}(q_j)$;
- $commonFacet(T^{-1}(q_j), T^{-1}(q_k))$: returns the common facet of $T^{-1}(q_j)$ and $T^{-1}(q_k)$);
- $isBlocked(T^{-1}(q_j), F, p_i)$: checks whether a particular facet F of $T^{-1}(q_j)$ is blocked in the sense that no trajectories can flow out of $T^{-1}(q_j)$ through this facet; and
- $isTransient(q_j, p_i)$: checks whether q_j is transient for mode p_i.

Algorithm 1 Abstraction Algorithm

Initialize $\mathcal{S} = \mathcal{Q}$, $\mathcal{S}_0 = \mathcal{Q}_0$, and $\xrightarrow{p} = \mathcal{S} \times \mathcal{S}$
for $i = 1 : |\mathcal{P}|$ **do**
  for $j = 1 : |\mathcal{Q}|$ **do**
    for $k = 1 : |\mathcal{Q}|$ **do**
      if $q_k \in Neighbors(T^{-1}(q_j))$ **then**
        $F = commonFacet(T^{-1}(q_j), T^{-1}(q_k))$
        if $isBlocked(T^{-1}(q_j), F, p_i)$ **then**
          $\xrightarrow{p_i} = \xrightarrow{p_i} \setminus (q_j, q_k)$
        end if
      else
        $\xrightarrow{p_i} = \xrightarrow{p_i} \setminus (q_j, q_k)$
      end if
    end for
    if $isTransient(q_j, p_i)$ **then**
      $\xrightarrow{p_i} = \xrightarrow{p_i} \setminus (q_j, q_j)$
    end if
  end for
end for
return $\{(\mathcal{S}, \mathcal{S}_0, \xrightarrow{p}) : p \in \mathcal{P}\}$

Algorithm 1 is generic in the sense that it can build upon existing results. In the literature, there has been considerable amount of work on reachability analysis for systems with specific dynamics, e.g., affine systems (e.g. Habets *et al.* [14] multi-affine systems (e.g. Belta and Habets [5]), and nonlinear systems (e.g. Girard and Martin [13] and Ben Sassi and Girard [6]). Some of these results can be adapted and applied to compute over-approximations using Algorithm 1 above, in particular, for evaluating the functions $isBlocked(T^{-1}(q_j), F, p_i)$ and $isTransient(q_j, p_i)$. Once such a finite-state approximation is obtained, the results in this paper can be used to synthesize switching controllers not only for reachability, but also for richer specifications expressible in LTL.

Remark 1: When all f_p's are polynomial vector fields and all $T^{-1}(q)$'s are semialgebraic, we can use sum-of-squares-based convex optimization to search for a function B satisfying the conditions of Proposition 1 in a principled and efficient way. This will provide computationally efficient means for evaluating $isTransient(q_j, p_i)$ in Algorithm 1. Similar techniques can be used to find certificates for $isBlocked(T^{-1}(q_j), F, p_i)$. The readers are referred to [30] for more details.

B. Product Transition System

Given an over-approximation $\mathcal{T}_q = \{(\mathcal{Q}, \mathcal{Q}_0, \xrightarrow{p}) : p \in \mathcal{P}\}$ for the switched system (2) and an environment transition system $\mathcal{T}_e = (\mathcal{S}, \mathcal{S}_0, \rightarrow)$ for its environment, we can define a product transition system for describing the overall system behavior.

Definition 8: The *product transition system* is a finite transition system $\mathcal{T}_s := (\mathcal{S}, \mathcal{S}_0, \rightarrow)$ with states in $\mathcal{S} = \mathcal{Q} \times \mathcal{E} \times \mathcal{P}$, initial states $\mathcal{S}_0 = \mathcal{Q}_0 \times \mathcal{E}_0 \times \mathcal{P}_0$, and transition relation $\rightarrow \subseteq \mathcal{S} \times \mathcal{S}$ such that $(q, e, p) \rightarrow (q', e', p')$ if and only if either $e = e'$ and $q \xrightarrow{p} q'$, or $q = q'$ and $e \rightarrow e'$.

It is evident from the definition that $\mathcal{T}_s$ consists of two parts: $\mathcal{T}_q$ represents the plant dynamics under switching and $\mathcal{T}_e$ represents environment changes. Here we consider the product transition system with interleaving semantics. That is, at each time instant, either the environment state e or the plant state q changes but they do not change simultaneously[1]. Defining the product transition system in an asynchronous way allows the switching mode to react in real time (in a continuous-time implementation) to environment changes. The price paid is the possibility of the undesired behavior that the environment $\mathcal{T}_e$ can make infinitely many transitions before $\mathcal{T}_q$ can make a transition and vice versa. However, as we assume environment signals are of finite variability, infinitely many transitions of the environment will require an infinite amount of time. On the other hand, due to underlying dynamics (1), we may show that certain states are transient under certain sets of modes, that is, the system cannot stay in that state indefinitely if we only switch within this given set of modes. Hence, the controller can ensure progress in a state, q, by restricting the modes to those on which q is transient. To encode such transience properties of a discrete state, we can augment the product transition system with some liveness assumptions expressible in $\mathrm{LTL}_{\setminus\bigcirc}$.

Proposition 2: If state q is transient on the family of modes $\mathcal{P}_s \subseteq \mathcal{P}$, then the following $\mathrm{LTL}_{\setminus\bigcirc}$ property is satisfied

$$\varphi_q := \neg\Diamond\Box\left(\left(\bigvee_{p\in\mathcal{P}_s} p\right) \wedge q\right). \tag{9}$$

[1]It is possible to consider a truly asynchronous execution to allow such behavior and the results extend to that case trivially.

Proof: It is clear from the semantics of LTL that formula (9) implies that the system cannot remain in state q forever, if the mode remains within the set $\mathcal{P}_s$, which is the definition of transience. ∎

Based on the product transition system $\mathcal{T}_s$ augmented with the liveness assumption φ_q in (9), we formulate the following discrete synthesis problem.

Discrete Switching Synthesis Problem: Given a product transition system $\mathcal{T}_s$ and the specification

$$\hat{\varphi} := ((\varphi_q \wedge \varphi_e) \rightarrow \varphi_s), \tag{10}$$

synthesize a (discrete) switching strategy that generates only correct executions in the sense that all executions satisfy $\hat{\varphi}$.

It will be shown that, by construction, our solution to the discrete synthesis problem can be continuously implemented to generate a solution for the continuous switching synthesis problem.

V. Discrete Synthesis as a Two-Player Game

In this section, we propose a two-player temporal logic game approach to the discrete switching synthesis problem formulated in the previous section. In particular, we recast the discrete synthesis problem as an infinite horizon game. We will first introduce the elements of the game structure.

A. Game Formulation

We consider two-player games, where, at each turn, a move of player 1 is followed by a move of player 2. We treat player 1 as adversarial, which tries to falsify the specification, whereas player 2 tries to satisfy it. Formally, a game is defined as follows.

Definition 9: [7] A *game structure* is a tuple

$$G = \langle \mathcal{V}, \mathcal{X}, \mathcal{Y}, \theta_e, \theta_s, \rho_e, \rho_s, \varphi \rangle,$$

where

- $\mathcal{V}$ is a finite set of *state variables*,
- $\mathcal{X} \subseteq \mathcal{V}$ is a set of *input variables*, i.e., variables controlled by player 1,
- $\mathcal{Y} = \mathcal{V} \setminus \mathcal{X}$ is the set of *output variables*, i.e., variables controlled by player 2,
- θ_e is an atomic proposition over $\mathcal{X}$ characterizing initial states of the input variables,
- θ_s is an atomic proposition over $\mathcal{Y}$ characterizing initial states of the output variables,
- $\rho_e(\mathcal{V}, \mathcal{X}')$ is the transition relation for player 1, which is an atomic proposition that relates a state and possible next input values (primed variables represent the value of a variable in the next step),
- $\rho_s(\mathcal{V}, \mathcal{X}', \mathcal{Y}')$ is the transition relation for player 2, which is an atomic proposition that relates a state and an input value to possible output values,
- ψ is the winning condition given by an LTL formula over $\mathcal{V}$.

The above definition gives a general game structure for LTL games. It is interpreted as follows in order to solve the discrete switching synthesis problem.

Game Structure for Switching Synthesis: Let $\mathcal{T}_s$ be the product transition system given by Definition 8, which consists of an over-approximation of (1) and an environment transition system, the game structure. We construct the game structure for solving the switching synthesis problem as follows:

- $\mathcal{V} = \{\mathbf{q}, \mathbf{e}, \mathbf{p}\}$, where $\mathbf{q}$ is the *plant* variable taking values in $\mathcal{Q}$, $\mathbf{e}$ the *environment* variable taking values in $\mathcal{E}$, $\mathbf{p}$ the *mode* variable taking values in $\mathcal{P}$;
- $\mathcal{X} = \{\mathbf{q}, \mathbf{e}\}$ and $\mathcal{Y} = \{\mathbf{p}\}$. In other words, both $\{\mathbf{q}, \mathbf{e}\}$ are controlled by player 1 and regarded as adversarial;
- θ_e defines the initial states of $\{\mathbf{q}, \mathbf{e}\}$ and θ_s captures that of $\{\mathbf{p}\}$;
- ρ_e and ρ_s are determined by the transition relation of $\mathcal{T}_s$, which encode both the non-deterministic transitions of the over-approximation $\mathcal{T}_q$ and the transitions of the environment allowed in $\mathcal{T}_e$; and
- the winning condition ψ is set to be $\hat{\varphi}$ in (10), i.e., the specification of the discrete synthesis problem.

We aim to solve this game in an efficient way, which will provide a solution to the discrete switching synthesis problem.

B. Switching Synthesis by Game Solving

While automatic synthesis of digital designs from general LTL specifications is one of the most challenging problems in computer science [7], for specifications in the form of the so-called Generalized Reactivity 1, or simply GR(1), formulas, it has been shown that checking its realizability and synthesizing the corresponding automaton can be accomplished in polynomial time in the number of states of the reactive system [7], [33].

GR(1) Game: A game structure defined in Definition 9 is a *GR(1) game* if the winning condition of the game structure is of the assume-guarantee form $\varphi := (\varphi_e \rightarrow \varphi_s)$ and, for $\alpha \in \{e, s\}$, φ_α is of the form

$$\varphi_\alpha := \bigwedge_{i \in I_1^\alpha} \Box \varphi_{1,i}^\alpha \bigwedge_{i \in I_2^\alpha} \Box \Diamond \varphi_{2,i}^\alpha, \tag{11}$$

where $\varphi_{1,i}^\alpha$ are propositional formulas characterizing safe, allowable moves, and invariants; and $\varphi_{2,i}^\alpha$ are propositional formulas characterizing states that should be attained infinitely often. Many interesting temporal specifications can be transformed into GR(1) specifications of the form (11). The readers can refer to [7], [33] for more precise treatment on how to use GR(1) game to solve LTL synthesis in many interesting cases (see also [44] for more examples).

Switching Synthesis as a GR(1) Game: Recall the assume-guarantee specification for the discrete synthesis problem:

$$\hat{\varphi} := ((\varphi_q \wedge \varphi_e) \rightarrow \varphi_s), \tag{12}$$

where φ_q is the liveness assumption in (9) encoding the transience properties in the product transition system $\mathcal{T}_s$, φ_e specifies a priori knowledge about the allowable environment behavior (beyond what is represented by the environment transition system), and φ_s describes the correct behavior of the overall system. While the liveness assumption φ_q is already of the form (11), we assume that, for $\alpha \in \{e, s\}$, each φ_α in (12) also has the structure in (11). Thus, we obtain a GR(1) game formulation for the switching synthesis problem, which can be solved efficiently.

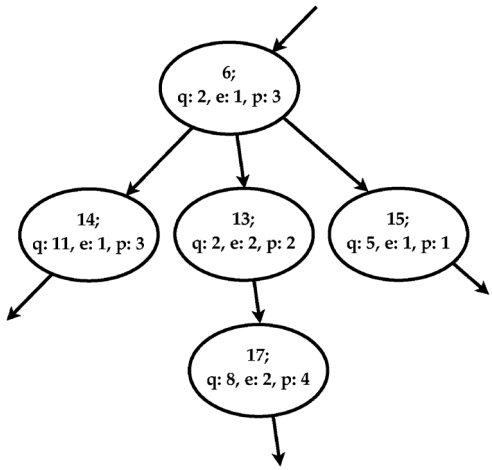


Fig. 2. Part of a synthesized automaton that can be used to extract a switching strategy. Suppose that the system arrives at state number 6, where $\mathbf{q} = 2$ and $\mathbf{e} = 1$. It chooses mode $\mathbf{p} = 3$ according to the automaton. Following this mode, if the environment does not change, i.e. $\mathbf{e} = 1$ is maintained, the system may end up at two different states $\mathbf{q} = 11$ and $\mathbf{q} = 5$ after a certain period of time, due to non-deterministic state transitions in the over-approximation. If the environment changes to $\mathbf{e} = 2$ before $\mathbf{q}$ changes, the system can react accordingly by possibly switching to a different mode (in this case $\mathbf{p} = 2$). Following the mode $\mathbf{p} = 2$, if $\mathbf{e}$ does not change, the system will end up in state number 17, where $\mathbf{q} = 8$ and $\mathbf{e} = 2$ and the system will switch to $\mathbf{p} = 4$.

Winning Strategy: A winning strategy for the system (player 2) is a partial function

$$(s_0 s_1 \cdots s_{t-1}, (q_t, e_t)) \mapsto p_t,$$

which chooses a switching mode based on the state sequence so far and the current moves of the environment and the plant such that formula (12) is satisfied. We say that φ is *realizable* if such a winning strategy exists. In addition, for two player LTL games, there always exists finite memory strategies that can be represented by finite automata. Thus, if the specification is realizable, solving the two-player game gives a finite automaton that encodes the winning strategy. A feedback switching protocol can be readily extracted from this winning automaton. More specifically, a mode switch by $\mathbf{p}$ is triggered only by a change of state of either $\mathbf{q}$ or $\mathbf{e}$. If there is no change in the environment, following a specific mode, the system will eventually go to a number of possible states that are allowed by the over-approximation. If there is a change in the environment, the system is allowed to react to this change immediately, i.e. without waiting until $\mathbf{q}$ changes its state. Thus, by observing both the plant and environment states, the next switching mode can always be chosen accordingly by reading the finite automaton. Fig. 2 shows a typical automaton given by solving a two-player game and how it is interpreted as a switching strategy for the system.

Remark 2: Given a two-player game structure and a GR(1) specification, the digital design synthesis tool implemented in JTLV [36] (a framework for developing temporal verification algorithm [7], [33]) generates a finite automaton that represents a switching strategy for the system. The Temporal Logic Planning (TuLiP) Toolbox, a collection of Python-based code for automatic synthesis of correct-by-construction embedded control software as discussed in [45] provides an interface to JTLV, which has been used for other applications [31], [32], [43], [45] and is also used to solve the examples later in this paper.

Remark 3: If there is no external environment and the over-approximation computed for (1), e.g., using Algorithm 1, happens to be deterministic in the sense that each of the transition systems in (6) is deterministic, the game synthesis problem reduces to a model checking problem [25]. In this sense, the temporal logic game approach outlined in this section covers model-checking based methods for controller synthesis as a special case [4], [11], [15], [20], [38]. There are certain trade-offs among computational complexity, conservatism in models and approximations, and expressivity of specifications, which may make one approach preferable to the other. On the one hand, model checking is amenable to highly-optimized software [10], [17], with computational complexity that is linear in the size of the state space [3], but it requires an approximation that needs to be deterministically implemented for all allowable exogenous disturbances. Such approximations are potentially difficult to obtain, in the sense that one may not be able to establish a sufficient number of (deterministic) transitions to make the problem solvable. On the other hand, over-approximations account for mismatch between the continuous model and its approximation as adversarial uncertainty and model it nondeterministically. Sufficient approximations of this type are potentially easier to establish and also allow us to further incorporate environmental adversaries, yet the resulting formulation is a two-player temporal logic game.

VI. Continuous Implementations of Reactive Switching Protocols

As discussed in the previous section, a (discrete) switching strategy, represented as a finite automaton, solves the discrete switching synthesis problem. In this section, we show that we can implement the (discrete) switching strategy at the continuous level, to provide a solution to the continuous switching synthesis problem we posed in Section III.

We define continuous implementations of executions of the product transition as follows. Let ρ be an execution of $\mathcal{T}_s$, i.e., an infinite sequence of triplets

$$\rho = (q, e, p) = (q_0, e_0, p_0)(q_1, e_1, p_1)(q_2, e_2, p_2) \cdots,$$

and $s = (x, \zeta, \sigma)$ a trajectory of the switched system (2) and its environment.

Definition 10: The trajectory s is said to be a *continuous implementation* of ρ, if there exists a sequence of non-overlapping intervals $\mathcal{I} = I_1, I_2, \cdots$ such that $\cup_{i=1}^{\infty} = \mathbb{R}^+$ and

$$x(t) \in T^{-1}(q_k), \zeta(t) = e_k, \sigma(t) = p_k, \forall t \in I_k, \forall k \geq 1.$$

Furthermore, if $l(I_k) \to \infty$ as $k \to \infty$, where $l(I_k)$ denotes the left end-point of I_k, the implementation is said to be *non-Zeno*.

Lemma 1: Let φ be an $\mathrm{LTL}_{\setminus\bigcirc}$ formula. If s is a non-Zeno implementation of ρ, then

$$s \models \varphi \quad \text{if and only if} \quad \rho \models \varphi. \tag{13}$$

Proof: The implementation is non-Zeno guarantees that s produces a well-defined word, say w_s. Let w_ρ be the word produced by ρ. We can easily show that w_s and w_ρ are stutter-

equivalent [3, Definition 7.86]. Therefore, we have $w_s \models \varphi$ if and only if $w_\rho \models \varphi$ [3, Theorem 7.92]. The statement (13) follows by definition. ∎

In this subsection, we show that every continuous implementation of the discrete strategy is provably correct in the sense that it only generates trajectories satisfying the given LTL specification, due to the way we construct the discrete abstraction of the continuous-time switched systems and solve the discrete synthesis problem. By exploiting properties of an over-approximation, we can show the following result.

Theorem 1: Given an over-approximation of (1), continuous implementations of the switching strategy obtained by solving a two-player game solve the continuous switching synthesis problem, if these implementations are non-Zeno.

Proof: By definition, the switching strategy given by solving a two-player game is a finite automaton that represents the winning switching strategy for the system. The automaton provides a switching mode for all possible moves of the environment and the plant. In view of Proposition 1, the theorem is proved, if we show that the switching strategy generates trajectories that continuously implement some execution, say $\rho = (q, e, p)$, of the winning automata. The semantics of the above switching strategy, applied to the switched system (2), are as follows. Given any initial state $x(0) \in T^{-1}(q_0)$ and an initial observation of the environment e_0, we choose $\sigma(0) = p_0$, by reading the automaton. From this point on, $x(t)$ evolves continuously following the dynamics of $\dot{x} = f_{p_0}(x, d)$ (subject to possible exogenous disturbances denoted by d). Environment signals are monitored in real time. If a change in environment is detected and this change does not violate the environment assumption, we update the switching mode immediately according to the automaton. If no environment changes are detected, by the definition of an over-approximation, there are two possibilities. First, $x(t)$ could stay in $T^{-1}(q_0)$ for all $t \geq 0$. In this case, we must have a self-transition in q for this particular mode. The sequence of intervals in Definition 10 can be chosen arbitrarily as long as they are non-overlapping (Zenoness can also be avoided by choosing the intervals that $l(I_k) \to \infty$ as $k \to \infty$). Second, there may exist some interval I_1 (before environment changes its states) and $q_1 \in \mathcal{Q}$ such that $x(t) \in T^{-1}(q_0)$ for all $t \in I_1$, and either (i) $x(\tau + \varepsilon) \in T^{-1}(q_1)$ for all sufficiently small $\varepsilon > 0$ and $\tau \in I_1$; or (ii) $x(\tau) \in T^{-1}(q_1)$ and $\tau \notin I_1$, where τ is a right end-point of I_1. In other words, the discrete state make a transition from q_0 to q_1 (while environment state remains the same). In either case, we can choose the next mode from the automata, by observing the evolution of the trajectory (among the two possibilities above). Note that even if an environment change is detected at the same time when $x(t)$ enters q_1 (although ruled out in our definition of product transition systems), we can update p in two separate steps. Once a next mode is generated, we repeat the same procedure as above. It is clear that the trajectory $x(t)$ generated this way implements one discrete execution of the winning automata. ∎

A. Discussions on Non-Zeno Implementations

We discuss in this subsection the possibility of Zeno behavior in the continuous implementations of a discrete strategy and propose appropriate assumptions to exclude such behavior. As discussed in Section V-B, a switching strategy given by solving a two-player game is a finite-state automaton. Each continuous implementation of the switching strategy correspond to an execution of this finite-state automaton, which is an infinite sequence of discrete states of the form (q, e, p).

Let $\mathcal{M}$ be a synthesized automaton. Consider simple cycles in $\mathcal{M}$, i.e., directed paths in $\mathcal{M}$ that start and end with the same state and no other repeated states in between. Let $\{\alpha_1, \cdots, \alpha_m\}$ denote the collection of simple cycles in $\mathcal{M}$ which include at least two states whose q-components are different, i.e., triplets (q_1, e_1, p_1) and (q_2, e_2, p_2) with $q_1 \neq q_2$. For each α_j, $j = 1, \cdots, m$, we can enumerate its states as $\{(q_i^j, e_i^j, p_i^j) : i = 1, \cdots, l_j\}$, where l_j is the length of the cycle α_j, and consider the set $\{q_1^j, \cdots, q_{l_j}^j\}$.

The following proposition can be used to rule out Zeno implementation of the strategy encoded in $\mathcal{M}$.

Proposition 3: If

$$\bigcap_{i=1}^{l_j} \overline{T^{-1}\left(q_i^j\right)} = \emptyset, \tag{14}$$

holds for $j = 1, \cdots, m$ where $\overline{A}$ indicate the closure of a set $A \subseteq \mathbb{R}^n$, and the environment signals are of finite variability, then all continuous implementations of the switching strategy given by $\mathcal{M}$ are non-Zeno.

Proof: Note that a Zeno implementation of $\mathcal{M}$ requires visiting an infinite number of discrete states within a finite time. This means that the trajectory should visit at least one simple cycle infinitely many times within a finite time. If the environment is of finite variability, this simple cycle should include at least two states with different q-components. Condition (14) excludes the possibility of visiting this simple cycle infinitely many times within a finite time (as completing one cycle would require a definite amount time). ∎

Remark 4: Since $T^{-1}(q) \cap T^{-1}(q') = \emptyset$ for $q \neq q'$, the above emptiness criterion is essentially on the boundaries of the cells $T^{-1}(q_i^j)$. Furthermore, if we can check that trajectories implementing the discrete transitions can only exit within a subset of the boundaries, such as the so-called exit set in [14], the above emptiness criterion for non-zenoness can be checked for subsets of the partitions $T^{-1}(q_i^j)$, which give more relaxed assumptions than (14). We also remark that even if the assumption is not satisfied for all possible executions of a given $\mathcal{M}$, Zenoness can still be avoided at the discrete level by recomputing abstractions such that the assumption holds, or by adding appropriate specifications to rule out executions whose periodic part can violate (14). Particularly, since the discrete states in the periodic part typically correspond to liveness specifications, by appropriately designing abstraction and specifying liveness properties, Zenoness can be avoided.

VII. Examples

A. Temperature Control

Consider a thermostat system [18] with four modes, ON, OFF, Heating, Cooling, as shown in Fig. 3, where the heating and cooling modes are included to capture what happens while

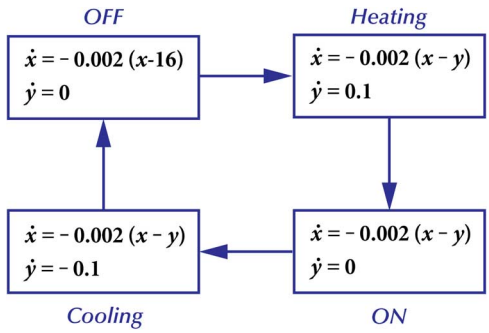


Fig. 3. Four-mode thermostat system.

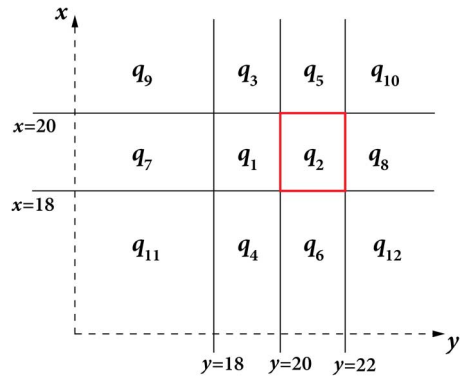


Fig. 4. Partition of the (x, y)-plane for the thermostat system for synthesizing a switching protocol that guarantees that the system reaches the region q_2.

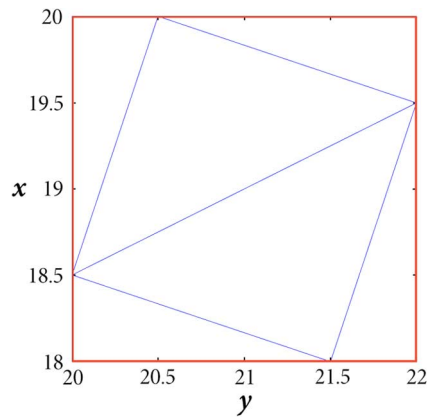


Fig. 5. Further partition of the region q_2 (indicated by the red square) in Fig. 4 for synthesizing a switching protocol that achieves invariance within the region q_2.

the heater is heating up to a desired temperature and cooling down to an allowable temperature, respectively. The dynamics of the four modes are also shown in Fig. 3, where x denotes the room temperature and y the temperature of the heater. In the OFF mode, the temperature changes at a rate proportional to the difference between the room temperature x and the outside temperature, which is equal to 16 in this case, according to Newton's law of cooling. In other modes, the change is at a rate proportional to the difference between the room temperature x and the temperature y of the heater. In the ON mode, the temperature of the heater is kept constant. In the heating mode, the temperature of the heater increases to 22 at a rate 0.1 per second; in the heating mode, the temperature of the heater decreases to 20 at a rate 0.1 per second.

We want the system to satisfy the following specifications:

(P1) Starting from any room temperature x and heater temperature y, the system has to reach a room temperature between 18 and 20 and a heater temperature between 20 and 22, i.e.,

$$\Diamond(18 \leq x \leq 20 \wedge 20 \leq y \leq 22).$$

(P2) If the room and heater temperatures are already in the desired range, they should stay so for all future time, i.e., the following requirement is enforced

$$\Box\left((18 \leq x \leq 20 \wedge 20 \leq y \leq 22)\right.$$
$$\left.\rightarrow \Box(18 \leq x \leq 20 \wedge 20 \leq y \leq 22)\right).$$

(P3) Transitions among the different modes have to be in the order shown in Fig. 3.

The specification consists of a reachability property and an invariance property in the (x, y)-plane, together with a sequential constraint in the modes. We start with the synthesis of a switching strategy that guarantees the reachability property. To obtain a proposition preserving abstraction, we partition the plane into 12 regions as shown in Fig. 4. The abstraction consists of plant variable q, whose states belong to $\mathcal{Q} = \{q_1, \cdots, q_{12}\}$, and a mode variable p, which takes values in $\mathcal{P} = \{N, F, H, C\}$, which represent the ON, OFF, Heating, and Cooling modes, respectively. By determining the transition relations among the regions in each mode, we obtain an over-approximation of the system in the sense of Definition 7.

To synthesize a switching protocol that realizes the reachability $\Diamond(q_2)$, we solve a two-player game as introduced in Section V-B. The switching protocol can be extracted from a finite automaton with 32 state.

We then consider the synthesis of a switching protocol that guarantees the invariance property, i.e., $q_2 \rightarrow \Box q_2$. For this purpose, we further partition q_2 into six subregions as shown in Fig. 5. We again obtain an over-approximation and solve a two-player game. The winning protocol can be extracted from a finite automaton with 8 states. A simulation result illustrating a continuous implementation of this protocol is shown in Fig. 6.

B. Automatic Transmission

Consider a 3-gear automatic transmission system [18] shown in Fig. 7. The longitudinal position of the car and its velocity are denoted by θ and ω, respectively. The transmission model has three different gears. For simplicity, the throttle position, denoted by u, takes value 1 in accelerating mode and -1 in decelerating modes. The specification concerns the efficiency of the automatic transmission and we use the functions

$$\eta_i(\omega) = 0.99 \exp\left(-\frac{(\omega - a_i)^2}{64}\right) + 0.01$$

to model the efficiency of gears $i = 1,2,3$, where $a_1 = 10$, $a_2 = 20$, $a_3 = 30$, as similarly considered in [18]. The acceleration in

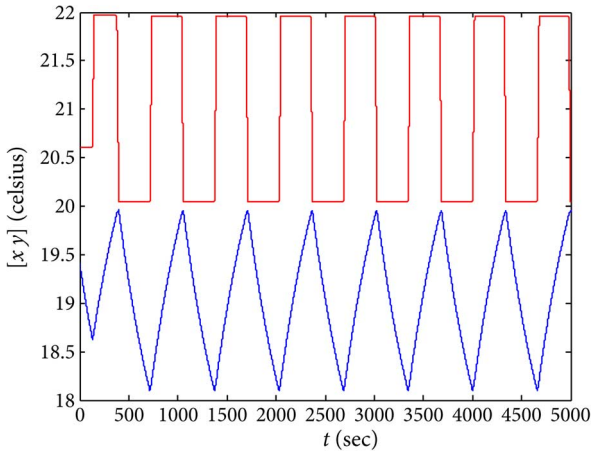


Fig. 6. Simulation illustrating the continuous implementation of a switching protocol that guarantees the invariance property $\square(18 \leq x \leq 20 \wedge 20 \leq y \leq 22)$: the red line represents the temperature y of the heater and blue line indicates the room temperature x.

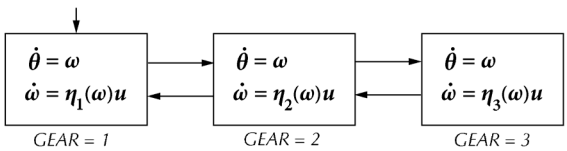


Fig. 7. A 3-gear automatic transmission.

mode i is given by the product of the throttle and transmission efficiency.

Let ζ be an environment signal taking value in $\{1,2\}$. The switching synthesis problem is to find a gear switching strategy to maintain a certain level of transmission efficiency when the speed is above certain value, while being reactive to the environment signal ζ. Formally, we consider a specification

$$\square(\omega \geq 5 \rightarrow \eta \geq 0.5) \wedge \square(0 \leq \omega \leq 40),$$

which consists of a minimum efficiency of 50% when the speed is greater than 5, and a speed limit of 40. In addition, the switching protocol is required to react to the continuous-time environment signal ζ by switching to a decelerating mode if $\zeta = 2$ and $\omega \geq 20$, i.e.,

$$\square((\omega \geq 20 \wedge \zeta = 2) \rightarrow (p = 2 \vee p = 4 \vee p = 6)).$$

Since the properties of interest here are related to ω only, we partition the ω-axis into a union of intervals $\mathcal{Q}$ that preserves the proposition on efficiency. The abstraction consists of a plant variable q, which takes values in $\mathcal{Q}$, and a mode variable p, which takes values in $\mathcal{P} = \{\pm 1, \pm 2, \pm 3\}$. Here, + and – denotes accelerating modes and decelerating modes, respectively. According to this abstraction, we have a deterministic transition system for each of the 7 modes. The switching synthesis problem can be solved by using the procedure outlined in Section V-B. The resulting switching controller can be represented by a finite automaton with 46 states. A simulation result is shown in Fig. 8, illustrating a continuous implementation of this strategy. The runtime reactiveness of the switching protocol is evident from the simulation.

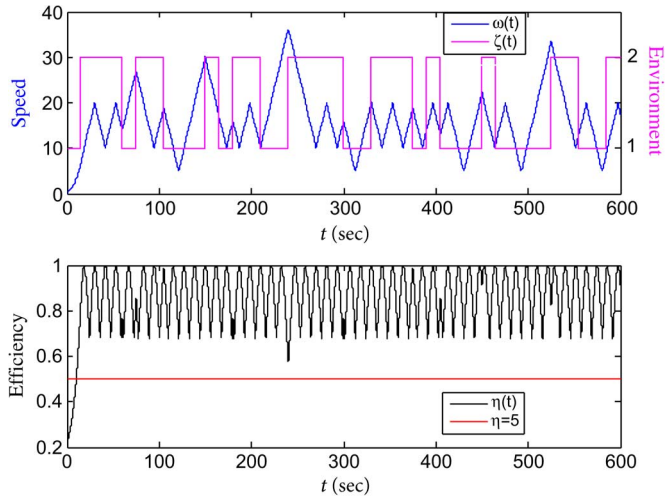


Fig. 8. Simulation results for Example 2: the upper figure shows the speed (ω) and environment (ζ) vs. time, while the lower figure shows the real-time efficiency (η) of the transmission, where the specified level 50% is indicated by the red line. It is evident from the upper figure that the system immediately switches to a decelerating mode if $(\omega \geq 20) \wedge (\zeta = 2)$ is satisfied.

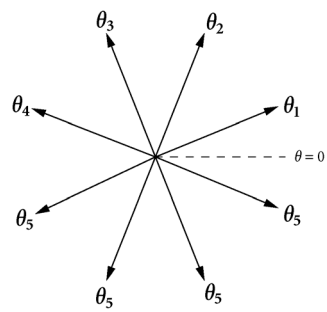


Fig. 9. Eight different heading angles of the robot.

C. Robot Motion Planning

Consider a kinematic model of a unicycle-type wheeled mobile robot [40] in 2D plane:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{15}$$

Here, x, y are the coordinates of the middle point between the driving wheels; θ is the heading angle of the vehicle relative to the x-axis of the coordinate system; v and w are the control inputs, which are the linear and angular velocity, respectively.

To cast the motion planning of this robot as a switching synthesis problem, we consider a situation where the heading angles are restricted to a finite set $\{\theta_p : p = 1, \cdots, 8\}$, where $\theta_p \in I_p$ and I_p are non-overlapping subintervals of $[0, 2\pi)$. Here we allow the heading angle to be within certain intervals to capture possible measurements errors or disturbances. The set of angles considered in this example is shown in Fig. 9, where θ_i can be an arbitrary angle in $((p-1)\pi/4, p\pi/4)$, for $p = 1, \cdots, 8$.

Equation (15) can now be viewed as a switched system with four different modes

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v_0 \cos\theta_p \\ v_0 \sin\theta_p \end{bmatrix}, \tag{16}$$

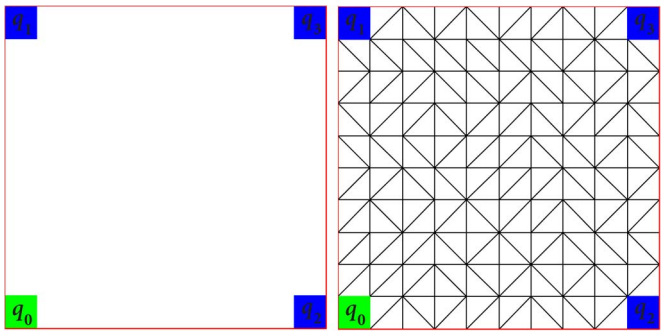


Fig. 10. Workspace for the Example 3 and its partition.

where $v_0 > 0$ is some constant speed. These dynamics can be achieved with inputs $(v, w) = (v_0, 0)$ in (15) with a desired heading angle in $\theta_p \in I_p$. In addition, we include a stop mode $p = 9$ with zero dynamics, i.e., $v_0 = w_0 = 0$ when $p = 9$. Transitions between different heading angles are now regarded as mode transitions, and the transition can be rendered through $\dot{x} = \dot{y} = 0$ and $\dot{\theta} = \omega_0$, by letting inputs $(v, w) = (0, w_0)$ in (15). In this sense, transitions can be made freely among different modes.

We consider a workspace shown on the left side of Fig. 10, which is a square of size 10. The robot is expected to satisfy the following desired properties:

(P1) Visit each of the blue cells, labeled as q_1, q_2, and q_3, infinitely often, and avoid obstacles.

(P2) Eventually go to the green cell q_0 after a PARK signal is received.

(P3) Stop whenever required.

Here, the PARK signal is an environment signal that constrains the behavior of the robot. The following assumption is made on the PARK signal.

(S1) Infinitely often, PARK signal is not received.

To synthesize a planner for this example, we introduce a partition of the workspace as shown on the right side of Fig. 10, in which each cell of size 1 is partitioned into two triangles. In each mode, we can determine the discrete transition relations according to Definition 7 and obtain an over-approximation of the system. Solving a two-player game as introduced in Section V-B gives a winning strategy that guarantees that the robot satisfies the given properties (P1)–(P3). Figs. 11 and 12 present simulation results under different situations. In the case of static obstacles with crossing objects, the resulting finite automaton of the switching protocol has 161 states, while it has 37628 states in the case of moving obstacles.

Remark 5: As all cells are transient under each of the modes in $\{1, \cdots, 8\}$, we encode these transience properties as additional assumptions as justified by Proposition 1. It is noted that, without these additional assumptions, the specifications in this example turn out to be unrealizable.

D. Numerical Example

We consider the polynomial dynamical system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 + u_1 + d_1 \\ x_1 + u_2 + d_2 \end{bmatrix} \quad (17)$$

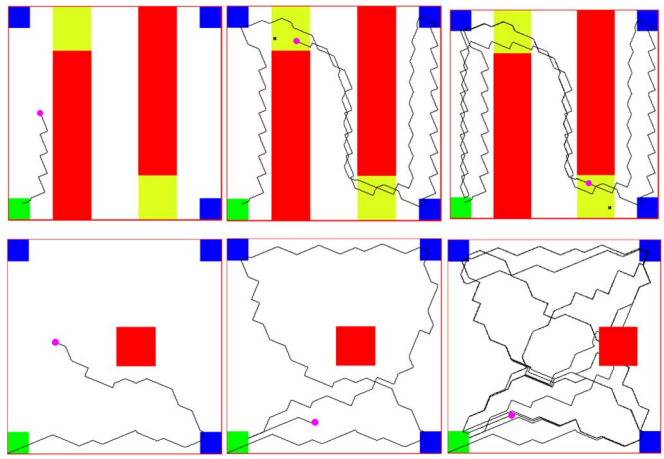


Fig. 11. Simulation results for Example 3. The upper figures show simulation results with static obstacles and possible crossing objects at intersections (marked as yellow regions). The robot is required to stop at intersections immediately if crossing objects (indicated by black crosses) are present. This requires runtime reactiveness of the switching protocol as further illustrated in Fig. 12. The lower figures show simulation results with a moving obstacle that occupies a square of size 2 and rambles horizontally under certain assumptions on its speed. The blue squares are the regions that the robot has to visit infinitely often. The green square is where the robot should eventually visit once a PARK signal is received. The obstacles are indicated by red, the trajectories of the robot are depicted by black curves, and the current positions of the robot are represented by the magenta dots.

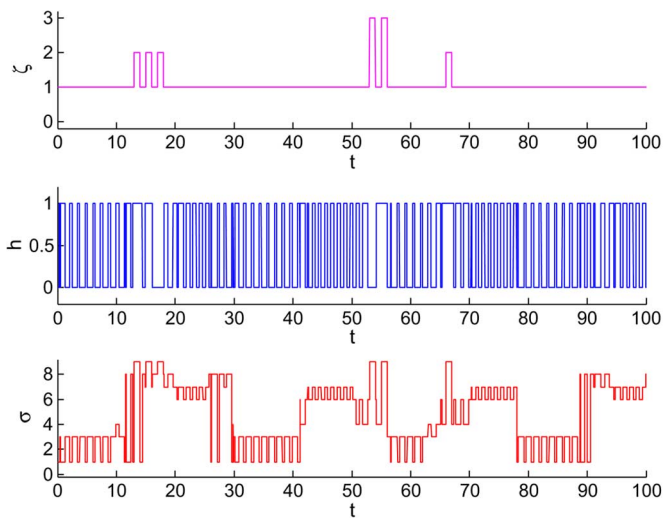


Fig. 12. Following Fig. 11, this figure shows the runtime reactiveness of the switching signal σ to changes in the environment for Example 3. Here ζ is an environment signal that indicates presence of crossing objects at intersections. In addition, h is a signal taking values in {0,1}, a change in whose value indicates that the robot enters a different cell (i.e., a change in q). It can be seen that in addition to reacting to changes in q, the switching signal also reacts immediately to changes in the environment signal ζ, by switching to the stop mode $p = 9$ in the presence of crossing objects at intersections (when $\zeta = 2,3$).

where $x = (x_1, x_2)$ lies in the domain $X = [-2, 2] \times [-1.5, 3]$ and the disturbance $d = (d_1, d_2)$ lies in $[-0.005, 0.005] \times [-0.005, 0.005]$. The control inputs are given by $u = (u_1, u_2) = K_p(x_1, x_2)$, $p \in \{1, 2, 3, 4\}$, which are four different state feedback controllers designed for this system, with two stabilizing it around desired equilibrium points, and the other two providing some fast dynamics within the region of interest X. In particular, the following controllers are used:

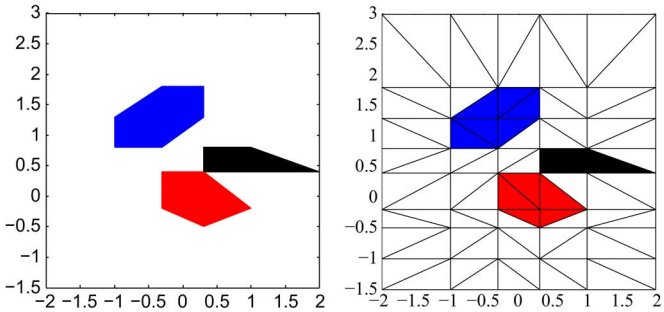


Fig. 13. Workspace for Example 4 and its partition.

$K_1(x) = (0, -x_2^2 + 2)$, $K_2(x) = (0, -x_2)$, $K_3(x) = (2, 10)$, and $K_4(x) = (-1.5, -10)$, where K_1 renders the point $(-0.669,\ 1.1540)$ as a locally stable equilibrium point, and K_2 renders the origin as a globally stable equilibrium point. These equilibrium points correspond to two desired operating conditions. The goal of the switching protocol is to safely steer the system in between these two. The domain, X, together with regions of interest, are shown on the left side of Fig. 13. Let A_1, A_2 and A_3 correspond to the red, blue and black regions, respectively. Formally, we would like to find a switching protocol such that the system satisfies the following specification:

(P1) Remain in X and visit the red and blue regions infinitely often, while avoiding the black region, i.e.,

$$\square X \wedge \square\neg A_3 \wedge \square\lozenge A_1 \wedge \square\lozenge A_2.$$

In addition, the switching protocol and the system is required to react to a continuous-time environment signal ζ taking value in {1,2} according to the following rules:

(P2.1) when $\zeta = 1$, the switching protocol is required to steer the system to the red region and stay there, without using the controller K_1; and

(P2.2) when $\zeta = 2$, the switching protocol is required to steer the system to the blue region and stay there, without using the controller K_2.

The following assumptions are made on the environment signal ζ:

(S1) ζ takes each of the values in {1,2} infinitely often; and

(S2) ζ changes its value only when the system is either in the red or blue region.

Based on a partition of the workspace as shown on the right side of Fig. 13, an over-approximation of system (17) can be computed using Algorithm 1 and sum-of-squares-based techniques as pointed in Remark 1. A switching protocol can be obtained by solving a two-player game as introduced in Section V-B. The resulting finite automaton that represents the switching protocol has 55 states. Simulation results, shown in Fig. 14, illustrate continuous implementations of the switching protocol, as well as its runtime reactiveness to the environment signal ζ.

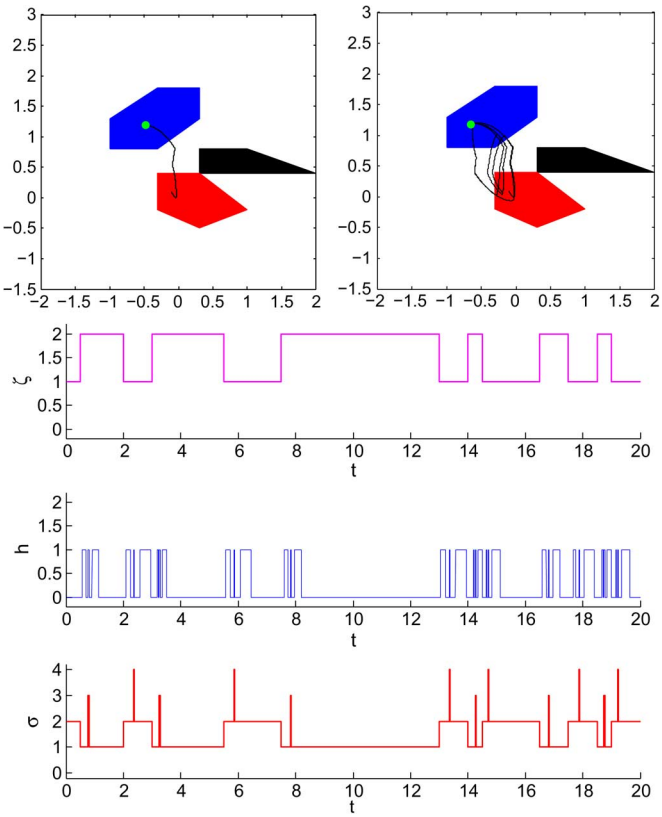


Fig. 14. Simulation results for Example 4. The upper figures show snapshots of a state trajectory in simulation. The blue and red regions are required to be visited infinitely often, while the black region is to be avoided. The lower three figures depict the switching signal σ, the environment signal ζ, and the changes in q (indicated by h, a signal taking values in {0,1}), as functions of time. It can be seen that σ reacts immediately to changes in ζ and q.

VIII. Conclusions

In this paper, we considered the problem of synthesizing switching protocols for nonlinear hybrid systems subject to exogenous disturbances. These protocols guarantee that the trajectories of the system satisfy certain high-level specifications expressed in linear temporal logic. We employed a hierarchical approach where the switching synthesis problem was lifted to discrete domain through finite-state abstractions. A family of finite-state transition systems, namely over-approximations, that abstract the behavior of the underlying continuous dynamical system were introduced. It was shown that the discrete synthesis problem for an over-approximation can be recast as a two-player temporal logic game and off-the-shelf software can be used to solve the resulting problem. Existence of solutions to the discrete synthesis problem guarantees the existence of continuous implementations that are correct by construction.

In contrast to existing work, we achieve runtime reactiveness of the continuous-time switching controller by explicitly modeling the environment and taking into account possibly asynchronous interactions between the system dynamics and the environment. We have accomplished this by defining product transition systems from the abstractions and augmenting these transition systems with additional liveness conditions to enforce progress in accordance with the underlying dynamics.

As discussed in the paper, there are certain trade-offs between the fidelity of the abstractions, expressiveness of the specifications that can be handled and computational complexity. To alleviate the latter issue and to further improve the scalability of the approach, future research directions include combining the results of this paper with a receding horizon framework and/or a distributed synthesis approach.

REFERENCES

[1] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proc. IEEE*, vol. 88, pp. 971–984, 2000.

[2] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli, "Effective synthesis of switching controllers for linear systems," *Proc. IEEE*, vol. 88, pp. 1011–1025, 2000.

[3] C. Baier and J. P. Katoen", *Principles of Model Checking*. Cambridge: MIT Press, 2008.

[4] G. Batt, C. Belta, and R. Weiss, "Temporal logic analysis of gene networks under parameter uncertainty," *IEEE Trans. Automat. Control*, vol. 53, pp. 215–229, 2008.

[5] C. Belta and L. Habets, "Controlling a class of nonlinear systems on rectangles," *IEEE Trans. Automat. Control*, vol. 51, pp. 1749–1759, 2006.

[6] M. A. Ben Sassi and A. Girard, "Computation of polytopic invariants for polynomial dynamical systems using linear programming," *Automatica*, vol. 48, pp. 3114–3121, 2012.

[7] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," *J. Comput. Syst. Sci.*, vol. 78, pp. 911–938, 2012.

[8] J. Cámara, A. Girard, and G. Gössler, "Synthesis of switching controllers using approximately bisimilar multiscale abstractions," in *Proc. HSCC*, 2011, pp. 191–200.

[9] A. Church, "Logic, arithmetic and automata," in *Proc. ICM*, 1962, pp. 23–35.

[10] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri, "NUSMV: A new symbolic model verifier," in *Proc. CAV*, 1999, pp. 495–499.

[11] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. ICRA*, 2005, pp. 2020–2025.

[12] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robotics*, vol. 21, pp. 1077–1091, 2005.

[13] A. Girard and S. Martin, "Control synthesis for constrained nonlinear systems using hybridization and robust controllers on simplices," *IEEE Trans. Automat. Control*, vol. 57, pp. 1046–1051, 2012.

[14] L. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Trans. Automat. Control*, vol. 51, pp. 938–948, 2006.

[15] L. Habets, M. Kloetzer, and C. Belta, "Control of rectangular multiaffine hybrid systems," in *Proc. CDC*, 2006, pp. 2619–2624.

[16] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Proc. CDC*, 1999, pp. 2655–2660.

[17] G. J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. New York: Addison-Wesley Professional, 2003.

[18] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Synthesizing switching logic for safety and dwell-time requirements," in *Proc. ICCPS*, 2010, pp. 22–31.

[19] M. Kloetzer and C. Belta, "Dealing with nondeterminism in symbolic control," in *Proc. HSCC*, 2008, pp. 287–300.

[20] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Automat. Control*, vol. 53, pp. 287–297, 2008.

[21] T. Koo, G. J. Pappas, and S. S. Sastry, "Mode switching synthesis for reachability specifications," in *Proc. HSCC*, 2001, pp. 333–346.

[22] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robotics*, vol. 25, pp. 1370–1381, 2009.

[23] J.-W. Lee and G. E. Dullerud, "Joint synthesis of switching and feedback for linear systems in discrete time," in *Proc. HSCC*, 2011, pp. 201–210.

[24] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Syst. Mag.*, vol. 19, pp. 59–70, 1999.

[25] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Switching protocol synthesis for temporal logic specifications," in *Proc. ACC*, 2012, pp. 727–734.

[26] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, 2004, pp. 71–76.

[27] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Berlin: Springer, 1992.

[28] T. Moor and J. Davoren, "Robust controller synthesis for hybrid systems using modal logic," in *Proc. HSCC*, 2001, pp. 433–446.

[29] A. S. Morse, "Supervisory control of families of linear set-point controllers--Part I: Exact matching," *IEEE Trans. Automat. Control*, vol. 41, pp. 1413–1431, 1996.

[30] N. Ozay, J. Liu, P. Prabhakar, and R. M. Murray, "Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems," in *Proc. ACC*, 2013.

[31] N. Ozay, U. Topcu, and R. M. Murray, "Distributed power allocation for vehicle management systems," in *Proc. CDC*, 2011.

[32] N. Ozay, U. Topcu, T. Wongpiromsarn, and R. M. Murray, "Distributed synthesis of control protocols for smart camera networks," in *Proc. ACM/IEEE Int. ICCPS*, 2011.

[33] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive (1) designs," in *Proc. VMCAI*, 2006, pp. 364–380.

[34] A. Pnueli, "The temporal logic of programs," in *Proc. FOCS*, 1977, pp. 46–57.

[35] A. Pnueli and R. Rosner, "On the synthesis of an asynchronous reactive module," in *Proc. ICALP*, 1989, pp. 652–671.

[36] A. Pnueli, Y. Sa'ar, and L. Zuck, "JTLV: A framework for developing verification algorithms," in *Proc. CAV*, 2010, vol. 6174, pp. 171–174 [Online]. Available: http://jtlv.ysaar.net/

[37] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. HSCC*, 2004, pp. 271–274.

[38] P. Tabuada and G. J. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Proc. HSCC*, 2003, pp. 498–513.

[39] A. Taly and A. Tiwari, "Switching logic synthesis for reachability," in *Proc. ACM Int. EMSOFT*, 2010, pp. 19–28.

[40] J. M. Toibero, F. Roberti, and R. Carelli, "Stable contour-following control of wheeled mobile robots," *Robotica*, vol. 27, pp. 1–12, 2009.

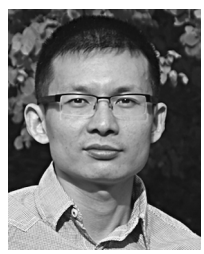
[41] J. Tumova, B. Yordanov, C. Belta, I. Cerna, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *Proc. CDC*, 2010, pp. 4230–4235.

[42] G. Weiss and R. Alur, "Automata based interfaces for control and scheduling," in *Proc. HSCC*, 2007, pp. 601–613.

[43] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Formal synthesis of embedded control software: Application to vehicle management systems," in *Proc. AIAA Infotech@Aerospace Conf.*, 2011.

[44] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Trans. Automat. Control*, vol. 57, pp. 2817–2830, 2012.

[45] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: A software toolbox for receding horizon temporal logic planning," in *Proc. HSCC*, 2011, pp. 313–314.



Jun Liu (M'11) received the B.S. degree in applied mathematics from Shanghai Jiao-Tong University, Shanghai, China, in 2002, the M.S. degree in mathematics from Peking University, Beijing, China, in 2005, and the Ph.D. degree in applied mathematics from the University of Waterloo, Waterloo, Canada, in 2010.

He is currently a Lecturer in the Department of Automatic Control and Systems Engineering at the University of Sheffield, Sheffield, U.K. Between 2011 and 2012, he was a Postdoctoral Scholar in Control and Dynamical Systems at the California Institute of Technology. His research interests are in the theory and applications of nonlinear, hybrid, and networked control systems.

Necmiye Ozay (M'04) received the B.S. degree from Bogazici University, Istanbul in 2004, the M.S. degree from the Pennsylvania State University, University Park, PA, USA, in 2006 and the Ph.D. degree from Northeastern University, Boston, MA, USA, in 2010, all in electrical engineering. Currently, she is a postdoctoral scholar at California Institute of Technology, Pasadena, CA, USA.

In Summer 2013, she will join the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA, as an assistant professor. Her research interests lie at the broad interface of dynamical systems, control, optimization and formal methods with applications in system identification and validation, autonomy and vision.

Dr. Ozay is the recipient of the 2008 IEEE Control Systems Society Conference on Decision and Control Best Student Paper Award and the 2009 IEEE Computer Society Biometrics Workshop Best Paper Honorable Mention Award.

Ufuk Topcu (M'08) received the Ph.D. degree in 2008 from the University of California, Berkeley, CA, USA and was a Postdoctoral Scholar at the California Institute of Technology, Pasadena, CA, USA, between 2008 and 2012.

He is a Research Assistant Professor at the University of Pennsylvania, Philadelphia, PA, USA. His research interests are on the analysis, design, and verification of networked, information-based systems with projects in autonomy, advanced air vehicle architectures, and energy networks.

Richard M. Murray (F'04) received the B.S. degree in electrical engineering from California Institute of Technology (Caltech), Pasadena, CA, USA, in 1985 and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1988 and 1991, respectively.

He is currently the Thomas E. and Doris Everhart Professor of Control & Dynamical Systems and Bioengineering at Caltech. His research is in the application of feedback and control to networked systems, with applications in biology and autonomy. Current projects include analysis and design biomolecular feedback circuits; specification, design and synthesis of networked control systems; and novel architectures for control using slow computing.