2012 American Control Conference
Fairmont Queen Elizabeth, Montréal, Canada
June 27-June 29, 2012

Switching Protocol Synthesis for Temporal Logic Specifications

Jun Liu, Necmiye Ozay, Ufuk Topcu, and Richard M. Murray

Abstract— **We consider the problem of synthesizing a robust switching controller for nonlinear hybrid systems to guarantee that the trajectories of the system satisfy a high level specification expressed in linear temporal logic. Two different types of finite transition systems, namely under-approximations and over-approximations, that abstract the behavior of the underlying continuous dynamical system are defined. Using these finite abstractions, it is possible to leverage tools from logic and automata theory to synthesize discrete mode sequences or strategies. In particular, we show that the discrete synthesis problem for an under-approximation can be reformulated as a model checking problem and that for an over-approximation can be transformed into a two-player game, which can then be solved by using off-the-shelf tools. By construction, existence of a discrete switching strategy for the discrete synthesis problem guarantees the existence of a continuous switching protocol for the continuous synthesis problem, which can be implemented at the continuous level to ensure the correctness of the trajectories for the nonlinear hybrid system. Moreover, in the case of over-approximations, it is shown that one can easily accommodate specifications that require reacting to possibly adversarial external events within the same framework.**

I. Introduction

The objective of this paper is synthesizing 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 [2], so-called motion primitives in robot motion planning [3], 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.

Specifically, given a family of system models, typically as ordinary differential equations potentially with bounded exogenous disturbances, and an LTL specification, our approach builds on a hierarchical representation of the system in each mode. The continuous evolution 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 evolution. Simulation-type relations [4] between the continuous and discrete models guarantee that the correctness of the synthesized switching protocols is preserved in the continuous implementation.

We consider two types of finite-state approximations for continuous nonlinear systems, namely under- and over-approximations. Roughly speaking, we call a finite transition system $\mathcal{T}$ an under-approximation if every transition in $\mathcal{T}$ can be continuously implemented for all allowable exogenous disturbances. In the case in which an under-approximation based finite-state abstraction is used, the switching protocol synthesis can be formulated as a model checking [5] problem. On the other hand, a finite transition system $\mathcal{T}$ is called an over-approximation if for each transition in $\mathcal{T}$, there is a possibility (due to either the exogenous disturbances or the coarseness of the approximation) for continuously implementing the strategy. We account for the mismatch between the continuous model and its over-approximation as adversarial uncertainty and model it nondeterministically. Consequently, the corresponding switching protocol synthesis problem is formulated as a two-player temporal logic game (see [6] and references therein and the pioneering work in [7]). This game formulation also allows us to incorporate adversarial environment variables that do not affect the dynamics of the system but constrain its behavior through the specification.

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). Jha *et al.* [8] focuses on switching logics that guarantee the satisfaction of certain safety and dwell-time requirements. Taly and Tiwari [9], Cámara *et al.* [10], Asarin *et al.* [11], and Koo *et al.* [12] consider a combination of safety and reachability properties. Joint synthesis of switching logics and feedback controllers for stability are studied by Lee and Dullerud [13]. The work by Frazzoli *et al.* [3] 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 [14].

The main contributions of the current paper are in extending the family of systems and specifications in switching protocol synthesis. The proposed methodology is applicable to a large family of system models potentially with exogenous disturbances along with an expressive specification language

This work was supported in part by the NSERC of Canada, the Multiscale Systems Center, and the Boeing Corporation. A full length version of this document is available at [1].

The authors are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA. {liu, necmiye, utopcu, murray}@cds.caltech.edu

(LTL in this case). The use of LTL enables to handle a wide variety of specifications beyond mere safety and reachability, as well as to account for potentially adversarial, a priori unknown environments in which the system operates (and therefore its correctness needs to be interpreted with respect to the allowable environment behaviors). Furthermore, the methodology improves the flexibility of switching protocol synthesis by merging ideas from multiple complementing directions and offering options that trade computational complexity with conservatism (and expressivity). For example, the resulting problem formulation with under-approximations of continuous evolution is amenable to highly-optimized software for model checking [15], [16], yet at the expense of increased conservatism in modeling. On the other hand, over-approximations are potentially easier to establish, yet the resulting formulation is a two-player temporal logic game (with publicly available solvers [17], [6] that are less evolved compared to the currently available model checkers). Another trade-off is in the family of two-player games considered here. Such games with complete LTL specifications is known to have prohibitively high computational complexity [18]. Therefore, we focus on an expressive fragment of LTL, namely Generalized Reactivity (1), with favorable computational complexity [6].

II. Preliminaries and Problem Formulation

A. Continuous-time switched systems

Consider a family of nonlinear systems,

$$\dot{x} = f_p(x, d), \quad p \in \mathcal{P}, \tag{1}$$

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 the 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

$$\dot{x} = f_\sigma(x, d), \tag{2}$$

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 [2]. We emphasize that, although the above formulation does not explicitly include a control input in its formulation, it can capture different situations where control inputs can be included, e.g., within each mode p, we may either assign a constant valued control input u_p, which can further belong to a finite number of quantized levels $\left\{u_p^1, u_p^2, \cdots, u_p^{L_p}\right\} \subseteq \mathbb{R}^m$, or choose a feedback controller $u(t) = K_p(x(t))$. Depending on different applications, each mode in (1) may represent, for example, a control component [14], [19], a motion primitive (which belongs to, e.g., a finite library of flight maneuvers [3], or a set of pre-designed behaviors [20]), and, in general, an operating mode of a multi-modal dynamical system [8], [9]. To achieve complex tasks, it is often necessary to compose these basic components. The composition can be enforced at a high-level control layer by implementing a switching protocol for mission-level specification. Designing correct switching protocols, however, can be a challenging issue [8], [11], [12], [14].

The goal of this paper is to propose methods for automatically synthesizing 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. LTL is a rich specification language that can express many desired properties, including safety, reachability, invariance, response, and/or a combination of these [21] (see also [22] for examples).

B. Problem description and solution strategy

Before formally stating the problem, we present a schematic description of the problem and its solution approach. The problem can be described as: given a family of system models in (1) and its specification expressed in LTL, synthesize a switching control protocol that, by construction, guarantees that the system satisfies its specification for all allowable exogenous disturbance. Within the same formulation, we also aim to incorporate environmental adversaries, which do not directly impact the dynamics of the system but constrain its behavior through the specification, and synthesize effective switching controllers for all valid environment behaviors. The solution of this problem enables us, e.g., to compose available controllers, which are predesigned to meet certain specifications, to achieve a high-level specification, as illustrated in Figure 1.

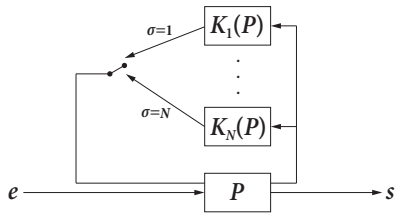

Fig. 1: P represents a plant subject to exogenous disturbances, $\{K_i(P) : i = 1, \cdots, N\}$ is a family of controllers, s represents the overall system behavior, e represents environmental adversaries, which do not directly impact the dynamics of the system but constrain its behavior through the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

Based on the continuous-time nonlinear system model (1), our hierarchical approach to the switching synthesis problem consists of two steps:

(i) We first establish finite-state approximations of the family of systems (1), which are a family of finite transition systems that approximate the dynamics in each mode.

(ii) We then synthesize a switching protocol based on high-level, discrete abstraction that, when continuously imple-

mented, ensures the correctness of the trajectories of the resulting switched system (2).

More specifically, we formulate two different types of discrete abstractions, namely *under-approximation* and *over-approximation*, respectively. For an under-approximation, the synthesis of a switching protocol is formulated as an LTL model checking [5] problem, which is amenable to highly optimized software implementations [15], [16]. For an over-approximation, we formulate the 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 [18], we restrict ourselves to an expressive fragment of LTL, namely Generalized Reactivity (1), with favorable computational complexity [6].

While exogenous disturbances are accounted for in the continuous level, adversarial environment behaviors are diverse and not necessarily amenable to modeling as an ordinary differential equation. Therefore, we defer the formal introduction of environment variables to Section III-B, where a two-player game formulation allows us to incorporate adversarial environment variables that do not affect the continuous-level dynamics of the system but rather constrain its behavior through the high-level specification.

C. Finite-state approximations

To formally state the synthesis problem, we define two types of finite-state abstractions of the continuous evolution in (1) and introduce the specification language LTL. LTL formulas are built upon a finite number of atomic propositions. An *atomic proposition* is a statement on system variables of interest that has a unique truth value (`True` or `False`) for a given value (called *state*) of each system variable. To formulate the switching synthesis problem, we are at least interested in two types of variables: the plant variable x and the switching mode variable p. Let $\Pi := \{\pi_1, \pi_2, \cdots, \pi_n\}$ be a set of atomic propositions. For example, each proposition $\pi_i \in \Pi$ can represent a domain in $\mathbb{R}^n$ and a set of modes in $\mathcal{P}$ of interest. Formally, for system (2), we associate an observation map $h : \mathbb{R}^n \times \mathcal{P} \to 2^{\Pi}$, which maps the continuous states and the discrete modes to a finite set of propositions. Without loss of generality, we consider h to be defined on the whole state space instead of some bounded invariant set. We also allow overlapping set of propositions since h is set-valued instead of single-valued.

Abstractions for each of the subsystems in (1) can be considered by defining an abstraction map $T : \mathbb{R}^n \to \mathcal{Q}$, which maps each state $x \in \mathbb{R}^n$ into a finite set $\mathcal{Q} := \{q_i : i = 1, \cdots, M\}$. The map T essentially defines a partition of the state space $\mathbb{R}^n$ by $\left\{T^{-1}(q) : q \in \mathcal{Q}\right\}$. We shall refer to elements in $\mathcal{Q}$ as discrete states of an abstraction. Finite-state approximations are defined in the following.

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

Consider a family of finite transition systems

$$\left\{\mathcal{T}_p := (\mathcal{Q}, \mathcal{Q}_0, \xrightarrow{p}) : p \in \mathcal{P}\right\}. \tag{3}$$

Definition 2: The family of finite transition systems in (3) is said to be an *under-approximation* of (1) if the following two statements hold.

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

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

satisfy

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

(ii) For any $q \in \mathcal{Q}$, if there is a self-transition $q \xrightarrow{p} q$, then, for all $x_0 \in T^{-1}(q)$ and all exogenous disturbances $d : [0, \infty) \to D \subseteq \mathbb{R}^d$, trajectories ξ of the pth subsystem of (1) starting from x_0 satisfy $\xi(t) \in T^{-1}(q), \quad \forall t \in [0, \infty)$, i.e., $T^{-1}(q)$ is a positively invariant set for the pth subsystem under all exogenous disturbances.

Definition 3: The family of finite transition systems in (3) 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 satisfies

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

(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 in $\mathcal{T}_p$ as long as there is a possibility (either induced by disturbances or a coarse partition) for the continuous system to implement the transition, whereas, in an under-approximation, a discrete transition $q \xrightarrow{p} q'$ is included in $\mathcal{T}_p$ only if the continuous flow can strictly implement the transition. In other words, an under-approximation includes only transitions that can be implemented by the continuous dynamics and an over-approximation includes all possible transitions.

In both approximations, time is abstracted out in the sense that we do not care how much time it takes to reach one discrete state from another. As the focus of this paper is on the automatic synthesis of switching protocols, we shall assume that we are given or we can construct a finite abstraction of the subsystems in (1), which is either an under-approximation or an over-approximation by Definitions 2 and 3.

For the above finite approximations to be consistent with continuous dynamics, they should preserve propositions of interest in the sense that for all x, $y \in \mathbb{R}^n$ and $p \in \mathcal{P}$,

$$T(x) = T(y) \Rightarrow h(x,p) = h(y,p), \tag{4}$$

where h is the observation map defined earlier. In other words, if two continuous states belong to the same subset of the continuous state space corresponding to the same discrete state in $\mathcal{Q}$, they should map to the same propositions under h.

D. LTL Syntax and Semantics

We use linear temporal logic (LTL) [23], [21] to formally specify system properties. 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).

Formally, given a set of atomic propositions Π, the set of LTL formulas over 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) $\varphi \rightarrow \psi := \neg\varphi \vee \psi$;
(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.

Continuous Semantics of LTL: An LTL formula for the continuous-time switching system (2) is interpreted over its trajectories (x, σ). Formally, given an LTL formula φ without the next operator $\bigcirc$, we can recursively define the satisfaction of φ over a trajectory $(x(t), \sigma(t))$ at time t, written $(x(t), \sigma(t)) \vDash \varphi$, as follows:

(1) for any atomic proposition $\pi \in \Pi$, $(x(t), \sigma(t)) \vDash \pi$ if and only if $\pi \in h(x(t), \sigma(t))$;
(2) $(x(t), \sigma(t)) \vDash \neg\varphi$ if and only if $(x(t), \sigma(t)) \nvDash \varphi$;
(3) $(x(t), \sigma(t)) \vDash \varphi \vee \psi$ if and only if $(x(t), \sigma(t)) \vDash \varphi$ or $(x(t), \sigma(t)) \vDash \psi$; and
(4) $(x(t), \sigma(t)) \vDash \varphi \mathcal{U} \psi$ if and only if there exists $t' \geq t$ such that $(x(t'), \sigma(t')) \vDash \psi$ and $(x(s), \sigma(s)) \vDash \varphi$ for all $s \in [t, t')$.

A trajectory (x, σ) starting at t_0 is said to satisfy φ, written $(x, \sigma) \vDash_{t_0} \varphi$, if $(x(t_0), \sigma(t_0)) \vDash \varphi$. If the initial time is not significant, we simply write $(x, \sigma) \vDash \varphi$.

Discrete Semantics of LTL: An LTL formula for a switched system given by the family of transition systems (3) is interpreted over its switching executions. Given an LTL formula φ, we can recursively define the satisfaction of φ over a switching execution $(q, p) = (q_0, p_0)(q_1, p_1)(q_2, p_2) \cdots$ at position i, written $(q_i, p_i) \vDash \varphi$, as follows:

(1) for any atomic proposition $\pi \in \Pi$, $(q_i, p_i) \vDash \pi$ if and only if there exists $x_i \in T^{-1}(q_i)$ such that $\pi \in h(x_i, p_i)$;
(2) $(q_i, p_i) \vDash \neg\varphi$ if and only if $(q_i, p_i) \nvDash \varphi$;
(3) $(q_i, p_i) \vDash \bigcirc\varphi$ if and only if $(q_{i+1}, p_{i+1}) \vDash \varphi$;
(4) $(q_i, p_i) \vDash \varphi \vee \psi$ if and only if $(q_i, p_i) \vDash \varphi$ or $(q_i, p_i) \vDash \psi$;
(5) $(q_i, p_i) \vDash \varphi \mathcal{U} \psi$ if and only if there exists $j \geq i$ such that $(q_j, p_j) \vDash \psi$ and $(q_k, p_k) \vDash \varphi$ for all $k \in [i, j)$.

A switching execution $(q, p) = (q_0, p_0)(q_1, p_1)(q_2, p_2) \cdots$ is said to satisfy φ, written $(q, p) \vDash \varphi$, if $(q_0, p_0) \vDash \varphi$.

E. Problem Formulation

Now we are ready to formally state our switching synthesis problems.

Continuous Switching Synthesis Problem: Given a family of continuous-time subsystems in (1) and a specification φ, synthesize a switching strategy that generates only correct trajectories (x, σ) in the sense that $(x, \sigma) \vDash \varphi$.

Discrete Switching Synthesis Problem: Given a family of finite transition systems in (3) and a specification φ, synthesize a switching strategy that generates only correct switching executions (q, p) in the sense that $(q, p) \vDash \varphi$.

We focus on the discrete synthesis problem and propose two different approaches depending on the types of abstractions in the sense of Definitions 2 and 3. It will be shown that, by construction, our solutions to the discrete switching synthesis problems from both approaches can be continuously implemented to generate a solution for the continuous switching synthesis problem.

III. Synthesis of Switching Protocols

In this section, we propose two approaches, one for each of the two types of finite-state approximations, to the discrete synthesis problem formulated in the previous section.

A. Switching synthesis by model checking

We start with the synthesis of switching protocol for an under-approximation of (1). Given such a finite approximation, the discrete synthesis problem can be reformulated as a model checking problem. Model checking [5], [24] is an automated verification technique that, given a finite-state model of a system and a formal specification, systematically checks whether this specification is satisfied. If not, the model checker provides a counterexample that indicates how the model could violate the specification. This counterexample is usually given as an execution path that violates the property being verified [5]. This execution path can either be finite, which leads from the initial system state to a single state that violates the property being verified, or be infinite, which leads to a loop of states, which is repeated infinitely many times and violates the property being verified. The counterexample being finite or infinite depends on the property being verified. Roughly speaking, a counterexample for safety and invariant properties is a finite path, while a counterexample for reachability and liveness properties is an infinite execution path [5].

Formally, to solve the switching synthesis problem by model checking, we construct a product transition system

$(\mathcal{Q}\times\mathcal{P}, \mathcal{Q}_0\times\mathcal{P}_0, \rightarrow)$ from the family of transition systems $\{\mathcal{T}_p\}$ in (3). Here $\mathcal{Q}\times\mathcal{P}$ is a set of system states that consist of switching modes $\mathcal{P}$ and plant states $\mathcal{Q}$, $\mathcal{Q}_0\times\mathcal{P}_0$ represents initial states, and $\rightarrow \subseteq (\mathcal{Q}\times\mathcal{P})\times(\mathcal{Q}\times\mathcal{P})$ is a transition relation: given states (q_i, p_i) and (q_j, p_j), there is a transition from (q_i, p_i) to (q_j, p_j) and we write $(q_i, p_i) \rightarrow (q_j, p_j)$, if $q_i \xrightarrow{p_i} q_j$, i.e., there exists a transition from q_i to q_j in the mode p_i.

We can solve a switching synthesis problem for a specification given by a temporal logic formula φ in the following procedure:
(1) Negate the formula φ to get $\neg\varphi$;
(2) Given the transition system $(\mathcal{Q}\times\mathcal{P}, \mathcal{Q}_0\times\mathcal{P}_0, \rightarrow)$ and the LTL formula $\neg\varphi$, determine if all executions of the transition system satisfy $\neg\varphi$.

The second step above is a model checking problem and can be solved by off-the-shelf software, e.g., the SPIN model checker [16] and the NuSMV symbolic model checker [15]. Solving this problem, there are two possible outcomes: (i) the model checker verifies that $\neg\varphi$ is true for the transition system $\mathcal{T}$; (ii) the model checker finds that $\neg\varphi$ is not true and provides a counterexample.

We are particularly interested in case (ii), since it provides a switching strategy that realizes φ and therefore solves our switching synthesis problem. Actually, a counterexample given by the model checker provides either a finite or infinite path of the form

$$(q_0, p_0) \rightarrow (q_1, p_1) \rightarrow (q_2, p_2) \rightarrow (q_3, p_3) \rightarrow \cdots \quad (5)$$

that violates the formula $\neg\varphi$, or in other words, satisfies φ. A switching strategy can be extracted from a counterexample found by model checking and given in the form (5).

Switching Strategy: Given a counterexample in the form (5), we consider two cases:
(i) if the path in (5) is infinite, we apply the switching sequence $p_0p_1p_2p_3\cdots$ to ensure that the execution

$$(q, p) = (q_0, p_0)(q_1, p_1)(q_2, p_2)(q_3, p_3)\cdots$$

satisfies φ;
(ii) if the path in (5) is finite and terminates at state (q_t, p_t), we apply any switching sequence with prefix $p_0p_1p_2p_3\cdots p_t$ to ensure that the execution

$$(q, p) = (q_0, p_0)(q_1, p_1)(q_2, p_2)(q_3, p_3)\cdots(q_t, p_t)\cdots$$

satisfies φ.

The switching protocol given by model checking is essentially an open loop strategy. It gives a mode sequence, by executing which the system is guaranteed to satisfy the specification. The correctness of the above switching strategy relies on the assumption that executions under the switching strategy can replicate the same state sequence as provided by the counterexample in the form (5). This assumption is implied if the family of transition systems (3) are an under-approximation of (1) in the sense of Definition 2. Formally, this is summarized in the following theorem.

Theorem 1: Given an under-approximation of (1), a switching strategy extracted from a counterexample found by model checking and given in the form (5) solves the discrete switching synthesis problem. This strategy can be continuously implemented to give a solution to the continuous switching synthesis problem, provided that $\{\mathcal{T}_p : p \in \mathcal{P}\}$ is an under-approximation of (1).

Remark 1: Based on an under-approximation, a model checker may verify that $\neg\varphi$ is true for the transition system $\mathcal{T}$. In such case, it does not necessarily mean that the switching synthesis problem does not have a solution. It could be the case that transition systems in (3), which serve as an abstract model for the underlying physical systems, are too crude for the switching synthesis problem to have a solution. A finer approximation may be needed for the discrete synthesis problem to be solvable.

B. Switching synthesis by game solving

In this subsection, we consider the case in which the family of transition subsystems in (3) are an over-approximation of (1). Our approach leverages recent work on reactive synthesis [6] of controllers for systems interacting with adversarial environments [22], where a control protocol is synthesized to generate a sequence of control signals to ensure that a plant meets its specification for all allowable behaviors of the environment. The synthesis problem is viewed as a two-player game between the environment and the plant: the environment attempts to falsify the specification and the plant tries to satisfy it.

We propose a temporal logic game approach to switching synthesis with an abstraction that gives an over-approximation. Due to nondeterminism inherent in an over-approximation, we may not be able to exactly reason about the discrete state transitions within each mode. Rather, we seek to construct mode sequences that can force the system to satisfy a given specification despite the nondeterminism of the state transitions in each mode. A game is constructed by regarding the discrete plant variable q as the *environment part*, which tries to falsify the specification, and a switching mode p as the *controllable part*, which tries to satisfy the specification. While automatic synthesis of digital designs from general LTL specifications is one of the most challenging problems in computer science [6], 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 [6].

We consider GR(1) specifications of the form $\varphi = (\varphi_q \rightarrow \varphi_s)$, where, roughly speaking, φ_q characterizes the nondeterministic transitions each subsystems can make, and φ_s describes the correct behavior of the overall switching system. Here, the non-deterministic transitions of the plant, specified in φ_q, are regarded as adversaries that try to falsify φ_s, while the switching mode is the controlled variable that tries to force the overall system to satisfy φ_s. We emphasize that, within the same framework, we can incorporate real environment into the system, by adding environment variables e that explicitly accounts for adversaries. Such adversaries do

not impact the continuous dynamics of the system directly, but rather constrain its behavior through GR(1) specifications of the form

$$\varphi = ((\varphi_q \wedge \varphi_e) \rightarrow \varphi_s), \tag{6}$$

where φ_e specifies allowable environment behaviors and φ_s is a system level specification that enforces correct behaviors for all valid environment behaviors. To be more precise, for $\alpha \in \{q, s, e\}$, each φ_α in (6) has the following structure:

$$\varphi_\alpha := \varphi^\alpha_{\text{init}} \wedge \bigwedge_{i \in I^\alpha_1} \Box \varphi^\alpha_{1,i} \wedge \bigwedge_{i \in I^\alpha_2} \Box \Diamond \varphi^\alpha_{2,i},$$

where $\varphi^\alpha_{\text{init}}$ is a propositional formula characterizing the initial conditions; $\varphi^\alpha_{1,i}$ are transition relations characterizing safe, allowable moves and propositional formulas characterizing invariants; and $\varphi^\alpha_{2,i}$ are propositional formulas characterizing states that should be attained infinitely often. Many interesting temporal specifications can be transformed into this form. The readers can refer to [6] for more precise treatment on how to use GR(1) game to solve LTL synthesis in many interesting cases (see also [22] for more examples). A winning strategy for the system, i.e., a strategy such that formula (6) is satisfied, can be solved by a symbolic algorithm within time complexity that is quadratic in the size of the state space [6].

We can formally describe our game approach for switching synthesis as follows.

Two-Player Game: A state of the game $s = (e, q, p)$ is in $\mathcal{E} \times \mathcal{Q} \times \mathcal{P}$, where $\mathcal{E}$, $\mathcal{Q}$, and $\mathcal{P}$ represent finite sets of environment states, plant states, and switching modes, respectively. A transition of the game is a move of the environment and a move of the plant, followed by a move of the switching mode. A switching strategy can be defined as 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. In this sense, a switching strategy is a winning strategy for the switching system such that the specification φ is met for all behaviors of the environment and the plant. We say that φ is *realizable* if such a winning strategy exists. If the specification is realizable, solving the two-player game gives a finite automaton that effectively gives a state-feedback switching protocol. More specifically, at each state, the system executes a switching mode, which drives the system to a number of possible states that are allowed in an over-approximation. By observing which state the system enters, the next switching mode is chosen accordingly by reading the finite automaton. By exploiting properties of an over-approximation, we can show the following result.

Theorem 2: Given an over-approximation of (1), a switching strategy obtained by solving a two-player game solves the discrete synthesis problem. This strategy can be continuously implemented to give a solution to the continuous switching synthesis problem, provided that $\{\mathcal{T}_p : p \in \mathcal{P}\}$ is an over-approximation of (1).

Remark 2: Given a two-player game structure and a GR(1) specification, the digital design synthesis tool implemented in JTLV [25] (a framework for developing temporal verification algorithm [6]) 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 [22], [26] provides an interface to JTLV, which has been used for other applications [22], [26]–[29] and is also used to solve the examples later in this paper.

Remark 3: Continuous implementations of a switching strategy may exhibit Zeno behavior, appropriate assumptions similar to that in [11] can be imposed to exclude such behavior. In particular, an emptiness criterion of the form $\bigcap_{i=1}^{l} \overline{T^{-1}(q_{f+i})} = \emptyset$, can be checked to rule out Zeno behavior. Here, the states q_{f+i} are from an execution of the switching strategy of the form

$$(q, p) = (q_0, p_0) \cdots (q_f, p_f) \Big((q_{f+1}, p_{f+1}) \cdots (q_{f+l}, p_{f+l}) \Big)^\omega,$$

where $l \in \mathbb{Z}^+$, and ω indicates a loop of states that are periodically repeated. More detailed discussions can be found in [1].

IV. Application to Robot Motion Planning

Consider a kinematic model of a unicycle-type wheeled mobile robot [20] in 2D plane:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \tag{7}$$

Here, x, y are the coordinates of the middle point tween 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 are $\{\theta_i : i = 1, \cdots, 8\}$, where each θ_i can be an arbitrary angle in $((i-1)\pi/4, i\pi/4)$, for $i = 1, \cdots, 8$.

Equation (7) can now be viewed as a switched system with eight different modes

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v_0 \cos\theta_p \\ v_0 \sin\theta_p \end{bmatrix}, \tag{8}$$

where $v_0 > 0$ is some constant speed. These dynamics can be achieved with inputs $(v, w) = (v_0, 0)$ in (7) with a desired heading angle in $\theta_p \in I_p$. 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 (7). In this sense, transitions can be made freely among different modes.

We consider a workspace shown on the left side of Figure 2, which is a square of size 10. The robot is expected to satisfy the following desired properties:

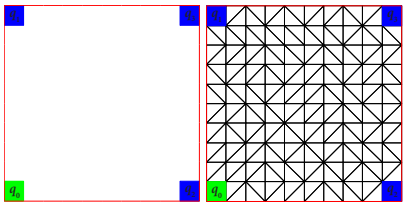

Fig. 2: The workspace for Example 3 and its partition.

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

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

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

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

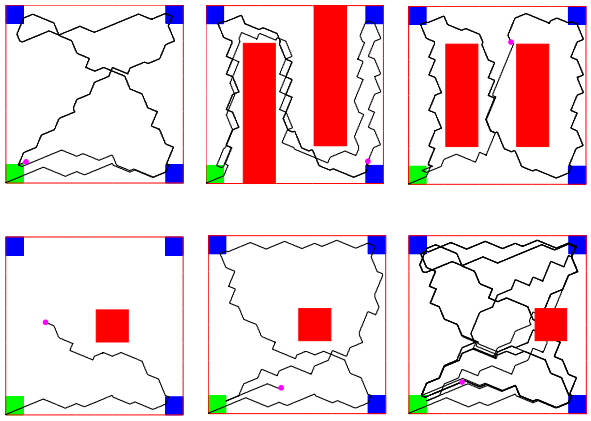

Fig. 3: Simulation results for Example 3: (a) The upper left figure shows simulation results without obstacles; (b) the upper middle and right figures show simulation results with different static obstacles; (c) the lower figures show simulation results with a moving obstacle that occupies a square of 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.

To synthesize a planner for this example, we introduce a partition of the workspace as shown on the right side of Figure 2, 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 3 and obtain an over-approximation of the system. Solving a two-player game as introduced in Section III-B gives a winning strategy that guarantees that the robot satisfies the given properties (P1) and (P2). In addition, we synthesize switching strategies for a workspace occupied with both static and moving obstacles. Snapshots of simulation results are shown in Figure 3, which illustrate continuous implementations of different switching strategies that are synthesized to achieve the specification under different situations with or without obstacles.

V. Conclusions and Discussions

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. Two different types of finite-state transition systems, namely under-approximations and over-approximations, that abstract the behavior of the underlying continuous dynamical system were introduced. It was shown that the discrete synthesis problem for an under-approximation led to a model checking problem. On the other hand, the discrete synthesis problem for an over-approximation was recast as a two-player temporal logic game. In both cases, off-the-shelf software can be used to solve the resulting problems. Moreover, existence of solutions to the discrete synthesis problem guarantees the existence of continuous implementations that are correct by construction.

This paper can be seen in the context of abstraction-based methods for controller synthesis and, in this sense, is closely related to existing work on construction of finite abstractions for nonlinear and hybrid systems (see [4] for an earlier review). Exact finite discrete abstractions, in the sense of bisimulation relations that require a one-to-one correspondence between system trajectories, are known to only exist for rather limited classes of systems [4]. Recent work therefore has focused on formulating relaxed notions of bisimulation relations, such as approximate and alternating bisimulation relations [30], [31]. In [32]–[34], it has been shown that approximate and approximate alternating bismulations can be obtained between a quantized control system and a finite transition system, if the underlying continuous-time nonlinear system is incrementally stable. In particular, the work by Girard *et al.* [32] focuses on incrementally stable switched systems. More recently, Zamani *et al.* [35] shows that such stability conditions can be further relaxed to incremental forward completeness. In general, these works focus on proving existence of approximate abstractions, and do not explicitly address the problem of controller synthesis for enforcing high-level specifications. Exceptions are [36] and [37], where, respectively, approximate bisimulations and approximate simulations are used to synthesize time-optimal controllers, which aim to steer, in minimal time, the state of the system to a desired target while remaining safe.

In this paper, we defined two types of abstractions, namely under- and over-approximations. While they resemble simulation and alternating simulation relations (as in [37]), respectively, they are based on the notion of language inclusion, which is in general a weaker notion than simulation, between the continuous-time systems and the discrete

transitions systems. This feature, by respecting linear time properties, is important in ensuring that the control strategies synthesized at the discrete level, when implemented continuously, can guarantee that the continuous-time systems satisfy certain LTL specifications. In this sense, our results are complementary to the aforementioned techniques and our contributions are twofold: (1) we formulated appropriate discrete approximations for systems with general nonlinear dynamics, rather than focusing on fully actuated or linear dynamics as considered in [22], [38]–[41]; (2) with the game formulation, we can explicitly account for non-determinism (regarded as adversarial) rather than lifting non-deterministic transition systems to deterministic ones [37], [42]. One advantage of doing so is to incorporate environmental adversaries within the same formulation, whereas none of the approaches mentioned above considers adversarial environment, with [40] as an exception. By restricting ourselves to the GR(1) fragment of LTL, the checking of realizability of the game and the synthesis of a discrete strategy have been shown to be of polynomial-time complexity [6]. Another advantage of considering non-deterministic approximations is that they are potentially easier to compute than deterministic approximations, by essentially allowing more flexibility in adding transitions (though non-deterministic) in the abstract systems. In the appendix of [1], we have discussed algorithms to apply or adapt existing techniques for computing both under- and over- approximations. Future work will focus on finding more efficient algorithms for computing such approximations for systems with general underlying dynamics.

REFERENCES

[1] J. Liu, N. Ozay, T. Ufuk, and R. Murray, "Synthesis of switching protocols from temporal logic specifications," Caltech, Tech. Rep., 2011.

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

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

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

[5] C. Baier and J. Katoen, *Principles of Model Checking*. MIT Press, 2008.

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

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

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

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

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

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

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

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

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

[15] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: a new Symbolic Model Verifier," in *Proc. of CAV*, 1999, pp. 495–499.

[16] G. Holzmann, *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2003.

[17] B. Jobstmann and R. Bloem, "Optimizations for ltl synthesis," in *Proc. of FMCAD*, 2006, pp. 117–124.

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

[19] D. Fisman and O. Kupferman, "Reasoning about finite-state switched systems," in *Proc. of the International Conf. on Hardware and Software: Verification and Testing*, 2011, pp. 71–86.

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

[21] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992, vol. 1.

[22] T. Wongpiromsarn, U. Topcu, and R. Murray, "Receding horizon temporal logic planning," *IEEE TAC, to appear*, 2012.

[23] A. Pnueli, "The temporal logic of programs," in *Proc. of the Annual Symp. on Foundations of Computer Science*, 1977, pp. 46–57.

[24] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 2000.

[25] A. Pnueli, Y. Sa'ar, and L. Zuck, "JTLV: A framework for developing verification algorithms," in *Proc. of CAV*, 2010, pp. 171–174.

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

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

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

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

[30] A. Girard and G. Pappas, "Approximation metrics for discrete and continuous systems," *IEEE TAC*, vol. 52, pp. 782–798, 2007.

[31] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach*. Springer-Verlag, 2009.

[32] A. Girard, G. Pola, and P. Tabuada, "Approximately bisimilar symbolic models for incrementally stable switched systems," *IEEE TAC*, vol. 55, pp. 116–126, 2010.

[33] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, pp. 2508–2516, 2008.

[34] G. Pola and P. Tabuada, "Symbolic models for nonlinear control systems: Alternating approximate bisimulations," *SIAM J. Control Optim.*, vol. 48, pp. 719–733, 2009.

[35] M. Zamani, G. Pola, M. Mazo Jr, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE TAC, to appear*, 2012.

[36] A. Girard, "Synthesis using approximately bisimilar abstractions: time-optimal control problems," in *Proc. of the IEEE CDC*, 2010, pp. 5893–5898.

[37] M. Mazo Jr and P. Tabuada, "Symbolic approximate time-optimal control," *Systems Control Lett.*, vol. 60, pp. 256–263, 2011.

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

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

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

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

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