Incremental Synthesis of Switching Protocols via Abstraction Refinement

Petter Nilsson and Necmiye Ozay

Abstract— **We consider the problem of synthesizing switching protocols that regulate the modes of a switched system in order to guarantee that the trajectories of the system satisfy certain high-level specifications. In particular, we develop a computational framework for incremental synthesis of switching protocols. Augmented finite transition systems are used as abstract representations of continuous dynamics. Inspired by counter-example guided abstraction refinement procedures for hybrid system verification, we start with a coarse abstraction and gradually refine it according to preorder relations on augmented finite transition systems. At each iteration, the proposed procedure can produce either a switching protocol that ensures the satisfaction of the specification, a certificate for nonexistence of such a protocol, or a refinement suggestion together with a partial solution to be used in the next iteration. Although the procedure is not guaranteed to terminate in general, we illustrate its practical applicability with two simple examples.**

I. Introduction

Designing a switching protocol that determines the mode signal of a switched system is crucial in many different applications. The modes of a switched system may represent different configurations of the system (e.g., corresponding to different valve or switch positions [1], [2]), the evolution of the system under various pre-designed feedback controllers tuned to achieve different performance criteria [3], [4], or, realization of different primitive tasks [5], [6]. The switching protocol orchestrates these low-level dynamics by identifying a mode signal in order to ensure that the trajectories of the system satisfy certain high-level specifications.

We consider reachability and safety specifications. Correct-by-construction controller synthesis from high-level specifications has attracted considerable attention in the past decade. Although a thorough survey is beyond the scope of this paper, we give a brief overview of the work that is most relevant to switching protocol synthesis. This line of research can be broadly classified into two categories: (i) synthesis methods based on direct computation of safe or reachable sets [1], [7], [8]; (ii) abstraction-based approaches that lift the problem to a finite space and solve the synthesis problem at the discrete-level [9], [10], [11], [2], [12]. Our method falls into this latter category. For linear [11] or incrementally stable [9] switched systems, it is possible to construct deterministic finite transition systems using (approximate) bisimulation relations. Alternatively, one can compute non-deterministic abstractions based on a finite set of predicates, which is applicable to a larger class of switched systems [10], [2], [12]. We do not make linearity or stability assumptions and consider non-deterministic abstractions. Instead of using a fixed set of predicates or a fixed state-space partition as in earlier work, we gradually refine the abstractions until either a switching protocol or a non-existence certificate is found; or the computational resources are exhausted.

Our approach can be seen as an extension of several methods in hybrid system verification to switching protocol synthesis. In particular, we are inspired by the counter-example guided abstraction refinement techniques [13], [14] and 3-valued abstractions [15]. We enhance the abstraction-refinement loop with an incremental synthesis algorithm. Incremental synthesis methods are gaining popularity, especially in robotic motion planning. Karaman and Frazzoli [16] employ incremental model checking for synthesis with deterministic abstractions, whereas we consider an incremental solution to a game as our abstract models are non-deterministic. Ulusoy *et al.* [17] and Livingston *et al.* [18] consider incremental synthesis in games with stochastic and non-deterministic agents, respectively. Both approaches start with finite discrete models, hence abstracting dynamics is not a concern in these papers. Our approach is also related to [19], where an incremental verification algorithm for discrete-time hybrid systems is proposed.

In [2], a general framework for synthesis of reactive switching protocols is proposed, where it is assumed that a finite description of the system can be computed. In [12] augmented finite transition systems serving as abstract models for switched systems are introduced, along with a preorder relation. Furthermore, a computationally tractable method for computing such abstract models given a partition of the state-space is proposed (see also [20]). In this paper, we further extend these earlier results by providing a computational framework for switching protocol synthesis where abstract models are automatically refined. The refinement is based on certificates from an incremental synthesis algorithm tailored for augmented finite transition systems. Integration of incremental synthesis and a refinement procedure mitigates the computational burden by systematically increasing the discrete state-space size only when needed. Another novelty of the current paper is that the proposed method can provide certificates for non-existence of switching protocols. Such certificates can be particularly useful to inform a low-level design process that additional modes, or new low-level feedback controllers, are needed for the system.

II. Problem formulation

In this section, we introduce the main components of the problem, namely continuous-time switched systems and

The authors are with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI, 48109. {pettni,necmiye}@umich.edu

augmented finite transition systems. We also define the switching protocol synthesis problem.

A. Continuous-time switched systems

A *continuous-time switched system* is a tuple $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a\in\mathcal{A}}, D)$, where the domain (i.e. state-space) $X \subset \mathbb{R}^n$ is a compact set, $\mathcal{A} \doteq \{1, \ldots, s\}$ is a finite index set counting the modes of the system, $\{f_a\}_{a\in\mathcal{A}}$ is a family of vector fields, and $D \subset \mathbb{R}^d$ is a compact disturbance set.

The evolution of the system $\mathcal{S}$ is governed by:

$$\dot{x}(t) = f_{\sigma(t)}(x(t), \delta(t)), \tag{1}$$

where $x(t) \in X \subset \mathbb{R}^n$ is the state, $\sigma(t) \in \mathcal{A}$ is the mode of the system, and $\delta(t) \in D \subset \mathbb{R}^d$ is the disturbance at time t. We assume that the switching signal $\sigma : \mathbb{R}^+ \to \mathcal{A}$ is piecewise constant with finite number of discontinuities on every bounded interval; and that δ and each f_a satisfy standard conditions to guarantee the existence and uniqueness of solutions of (1) when $\sigma(t) = a$ for all t. Given an initial condition $x(0) \in X$, a switching signal σ and a disturbance signal δ, they together define a unique *state trajectory* $x : [0, t_f] \to X$ that satisfies (1) for all $t \in [0, t_f]$, where $t_f \doteq \inf\{t : x(t) \notin X\}$.

The basic switching protocol synthesis problem we address in this paper is the following.

Problem 1: **[Continuous reach-avoid-stay game]** Given a switched system $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a\in\mathcal{A}}, D)$, a set $X_0 \subseteq X$ of initial states, a goal set $X_g \subseteq X$ and a bad set $X_b \subseteq X$, find a state-feedback mode signal σ such that for all $x(0) \in X_0$, the trajectories $x : [0, t_f] \to X$ of $\mathcal{S}$ satisfy the following:

1.1 the trajectories never leave $X \setminus X_b$ (implies $t_f = \infty$);
1.2 t_r defined as $t_r \doteq \inf\{t : x(t) \in X_g\}$ is finite, and $x(t) \in X_g$ for all $t > t_r$;

or, declare that no such mode signal exists.

A specific instance of a continuous reach-avoid-stay game is denoted by $\langle \mathcal{S}, (X_0, X_g, X_b)\rangle$. A mode signal ensuring satisfaction of conditions 1.1 and 1.2 is called a *winning switching protocol* for the game $\langle \mathcal{S}, (X_0, X_g, X_b)\rangle$. We also consider a variant of Problem 1, where the set X_0 is not specified (a game denoted by $\langle \mathcal{S}, (X_g, X_b)\rangle$) and the objective is to find a set $W^{(x)} \subseteq X$ such that for all initial conditions $x(0) \in W^{(x)}$, it is possible to find a mode signal so that conditions 1.1 and 1.2 are satisfied. Such a set $W^{(x)}$ is called a *winning set* for $\mathcal{S}$ in the game $\langle \mathcal{S}, (X_g, X_b)\rangle$.

B. Augmented finite transition systems

An *augmented finite transition system* is a tuple $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \mathcal{G})$ where Q is a finite set of states, $\mathcal{A}$ a finite set of actions (i.e., control inputs), $\to_{\mathcal{T}} \subseteq Q \times \mathcal{A} \times Q$ a transition relation, and $\mathcal{G} : \mathcal{A} \to 2^{2^Q}$ a progress group map, respectively. The progress group map $\mathcal{G}$ maps each action $a \in \mathcal{A}$ to a set of subsets of Q such that the system cannot remain indefinitely within any set of states $G \in \mathcal{G}(a)$ by using only the action a. A set $G \in \mathcal{G}(a)$ is called a *progress group under action* a. For each action $a \in \mathcal{A}$, there is an associated directed graph $T_a = (Q, \xrightarrow{a})$, where the set $\xrightarrow{a}$ of edges corresponds to the transitions in $\to_{\mathcal{T}}$ with action a. For an augmented finite transition system to be *well-formed*, it is required that for all $a \in \mathcal{A}$ and for all $G \in \mathcal{G}(a)$, given a state $q_1 \in G$, there exists a path from q_1 to some state $q_2 \notin G$ in the graph T_a. Note that, otherwise, the system cannot make progress from the state q_1 to a state not in G, which would contradict to the definition of progress group. An *execution* ρ of an augmented finite transition system $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \mathcal{G})$ is a sequence of pairs $\rho = (q(0), a(0))(q(1), a(1))(q(2), a(2))\ldots$, where $(q(i), a(i), q(i+1)) \in \to_{\mathcal{T}}$ for all $i \geq 0$ and ρ satisfies the progress conditions encoded by $\mathcal{G}$. A *state trajectory* $\rho|_q = q(0)q(1)q(2)\ldots$ of $\mathcal{T}$ is the projection of an execution ρ onto the set of states.

Finite transition systems considered in this paper are nondeterministic, that is, from a given state with a given action, there are multiple states the system can transition to. A *control strategy* for an augmented transition system $\mathcal{T}$ is a partial function $\mu : (q(0), a(0), \ldots, q(i-1), a(i-1), q(i)) \mapsto a(i)$ that maps the execution history to the next action.

Similarly to Problem 1, we now define reach-avoid-stay games for augmented finite transition systems.

Problem 2: **[Discrete reach-avoid-stay game]** Given a well-formed augmented finite transition system $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \mathcal{G})$, a set $Q_0 \subseteq Q$ of initial states, a goal set $Q_g \subseteq Q$ and a bad set $Q_b \subseteq Q$, synthesize a control strategy μ that generates state trajectories $q(0)q(1)q(2)\ldots$ such that whenever $q(0) \in Q_0$, the state trajectory satisfies the following:

2.1 for all $k \geq 0$, $q(k) \notin Q_b$;
2.2 k_r defined as $k_r \doteq \min\{k : q(k) \in Q_g\}$ is finite, and $q(k) \in Q_g$ for all $k \geq k_r$;

or, provide a certificate for non-existence of such a strategy.

A specific instance of a discrete reach-avoid-stay game is denoted by $\langle \mathcal{T}, (Q_0, Q_g, Q_b)\rangle$. A control strategy guaranteeing conditions 2.1 and 2.2 is called a *winning strategy* for the game $\langle \mathcal{T}, (Q_0, Q_g, Q_b)\rangle$. When Q_0 is not specified, one can define *the winning set* $W^{(q)}$ of the game $\langle \mathcal{T}, (Q_g, Q_b)\rangle$ as the set of all states from which there exists a control strategy that ensures the satisfaction of conditions 2.1 and 2.2.

When no strategy (resp. mode signal) that solves Problem 2 (resp. Problem 1) exists, the problem is said to be *unrealizable*.

III. Solution overview

The objective of this paper is to develop a computational framework to solve Problem 1. However, Problem 1 is generally undecidable with some exceptions when the dynamics (i.e., the vector fields f) are fairly simple [21]. On the other hand, Problem 2 constitutes a special case of two-player temporal logic games for which there are polynomial-time algorithms that can efficiently determine whether a winning strategy exists, and in that case compute one [22].

Given a continuous reach-avoid-stay game $\langle \mathcal{S}, (X_0, X_g, X_b)\rangle$ and a maximum iteration count K, we will incrementally generate a sequence of discrete reach-avoid-stay games $\left\langle \mathcal{T}^t, (Q_g^t, Q_b^t)\right\rangle_{t\in\{1,\ldots,K\}}$. The

generated $\mathcal{T}^t$'s will be gradually better (in a sense to be made clear next) approximations of $\mathcal{S}$ and winning sets of these discrete games will be used to reason about the solution of $\langle \mathcal{S}, (X_0, X_g, X_b) \rangle$.

A. Abstraction and refinement relations

We next introduce system relations adapted from [12], first relations between switched systems and augmented finite transition systems, and thereafter relations between different augmented finite transition systems.

In order to relate the sets of behaviors of a switched system $\mathcal{S}$ with those of a finite transition system $\mathcal{T}$, a common language to compare trajectories of $\mathcal{S}$ and $\mathcal{T}$ is required. For this purpose, a finite set Π of propositions is used. Accordingly, the switched system $\mathcal{S}$ is associated with an observation map $h_X : X \to 2^{\Pi}$ which gives an extended representation in the form of a tuple $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, D, \Pi, h_X)$. An augmented finite transition system $\mathcal{T}$ can also be extended with an observation map $h_Q : Q \to 2^{\Pi}$ to obtain a representation $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \Pi, h_Q, \mathcal{G})$. With these representations, it is possible to define relations on different systems sharing a common set Π of propositions.

We start with the definition of transience, an important property while reasoning about the trajectories of nonlinear switched systems. Then, we define a class of augmented finite transition systems that abstracts the behavior of a switched system.

Definition 1: Given a switched system $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, D, \Pi, h_X)$, a set $Y \subset X$ is *transient* on mode $a \in \mathcal{A}$ if and only if for any state $\xi_0 \in Y$ and for any disturbance signal δ taking values in the set D, there exists a finite time τ such that the solution of $\dot{x}(t) = f_a(x(t), \delta(t))$ leaves Y at time τ.

Definition 2: An augmented finite transition system $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \Pi, h_Q, \mathcal{G})$, is said to be an *over-approximation* for the switched system $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, D, \Pi, h_X)$, denoted by $\mathcal{T} \underset{\text{O.A.}}{\succeq} \mathcal{S}$, if there exists a function $\alpha : X \to Q$, such that the following statements hold.

(i) For all $\xi \in X$, $h_X(\xi) = h_Q(\alpha(\xi))$.

(ii) Given states $q, q' \in Q$, there is a transition $(q, a, q') \in \to_{\mathcal{T}}$, if there exist $\xi_0 \in \alpha^{-1}(q)$, a time $\tau > 0$, and some exogenous disturbance $\delta : [0, \tau] \to D$ such that the corresponding trajectory x of the subsystem f_a starting from ξ_0, i.e., $x : [0, \tau] \to \mathbb{R}^n$ with

$$x(0) = \xi_0, \quad \dot{x}(t) = f_a(x(t), \delta(t)), \quad \forall t \in (0, \tau),$$

satisfies

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

(iii) The progress group map $\mathcal{G}$ is such that given an action $a \in \mathcal{A}$, for all $G \in \mathcal{G}(a)$, the set $\bigcup_{q \in G} \alpha^{-1}(q)$ is transient on mode a of $\mathcal{S}$.

The next definition gives a relation between two augmented finite transition systems, which leads to a preorder.

Definition 3: Given two augmented finite transition systems $\hat{\mathcal{T}} = (\hat{Q}, \mathcal{A}, \to_{\hat{\mathcal{T}}}, \Pi, \hat{h}_{\hat{Q}}, \hat{\mathcal{G}})$ and $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \Pi, h_Q, \mathcal{G})$, $\mathcal{T}$ is said to be a *refinement* of $\hat{\mathcal{T}}$ (or, $\hat{\mathcal{T}}$ is an *abstract model* of $\mathcal{T}$), denoted by $\hat{\mathcal{T}} \underset{\text{A.S.}}{\succeq} \mathcal{T}$, if there exists a function $\beta : Q \to \hat{Q}$ such that the following conditions hold.

(i) For all $q \in Q$, $h_Q(q) = \hat{h}_{\hat{Q}}(\beta(q))$.

(ii) For all $(q_1, a, q_2) \in \to_{\mathcal{T}}$, $(\beta(q_1), a, \beta(q_2)) \in \to_{\hat{\mathcal{T}}}$.

(iii) For all $a \in \mathcal{A}$, for all $\hat{G} \in \hat{\mathcal{G}}(a)$, there exists $G \in \mathcal{G}(a)$ such that for all $\hat{q} \in \hat{G}$, we have $\beta^{-1}(\hat{q}) \subseteq G$.

The functions α in Def. 2 and β in Def. 3 are called *abstraction functions*.

Proposition 1: Let $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, D, \Pi, h_X)$, $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \Pi, h_Q, \mathcal{G})$, with $\Pi = \{\pi_g, \pi_b\}$, and h_X, h_Q be defined as follows:

- for all $i \in \{g, b\}$, $\pi_i \in h_X(\xi)$ if and only if $\xi \in X_i$;
- for all $i \in \{g, b\}$, $\pi_i \in h_Q(q)$ if and only if $q \in Q_i$,

for some sets $X_i \subseteq X$ and $Q_i \subseteq Q$. If $\mathcal{T} \underset{\text{O.A.}}{\succeq} \mathcal{S}$ and if $W^{(q)}$ is the winning set of the game $\langle \mathcal{T}, (Q_g, Q_b) \rangle$, then the set $\alpha^{-1}(W^{(q)}) \subseteq X$ is a winning set of the game $\langle \mathcal{S}, (X_g, X_b) \rangle$.

Proof: Noting that reach-avoid-stay objective can be expressed in linear temporal logic without next operator[1], the result directly follows from properties of overapproximations given in Proposition 3 in [12]. ∎

Corollary 1: Given $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a \in \mathcal{A}}, D, \Pi, h_X)$ and $\mathcal{T} = (Q, \mathcal{A}, \to_{\mathcal{T}}, \Pi, h_Q, \mathcal{G})$ with $\Pi = \{\pi_0, \pi_g, \pi_b\}$, and h_X and h_Q defined as follows:

- for all $i \in \{0, g, b\}$, $\pi_i \in h_X(\xi)$ if and only if $\xi \in X_i$;
- for all $i \in \{0, g, b\}$, $\pi_i \in h_Q(q)$ if and only if $q \in Q_i$,

for some sets $X_i \subseteq X$ and $Q_i \subseteq Q$, respectively. If $\mathcal{T} \underset{\text{O.A.}}{\succeq} \mathcal{S}$ and if there is a winning strategy in the game $\langle \mathcal{T}, (Q_0, Q_g, Q_b) \rangle$, then there is a winning switching protocol in the game $\langle \mathcal{S}, (X_0, X_g, X_b) \rangle$.

IV. Abstraction refinement loop

In this section we first summarize the overall abstraction refinement loop. Then, we elaborate on different pieces (i.e., algorithms) in the loop and argue their correctness.

A. Overview

Fig. 1 shows the proposed abstraction refinement loop. In order to solve Problem 1, we start by computing an augmented finite transition system $\mathcal{T}^0$ that is an over-approximation of $\mathcal{S}$. This step is explained in Section IV-B, using ideas borrowed from [12]. Then, we use the incremental synthesis algorithm described in Section IV-C to solve a discrete reach-avoid-stay game on $\mathcal{T}^0$. The synthesis algorithm outputs four sets of states: (i) winning set $W^{(q)}$, (ii) for-sure losing set $L^{(q)}$, (iii) a candidate winning set $C_W^{(q)}$ where abstraction refinement may enable expansion of the winning set, and, (iv) a candidate losing set $C_L^{(q)}$ where

[1]The linear temporal logic formula φ equivalent to the reach-avoid-stay game objective is given by $\varphi \doteq \pi_0 \to ((\neg\pi_b \wedge \neg\pi_g) \,\mathcal{U}\, \Box(\neg\pi_b \wedge \pi_g))$. We refer the interested reader to [23] for syntax and semantics of linear temporal logic.

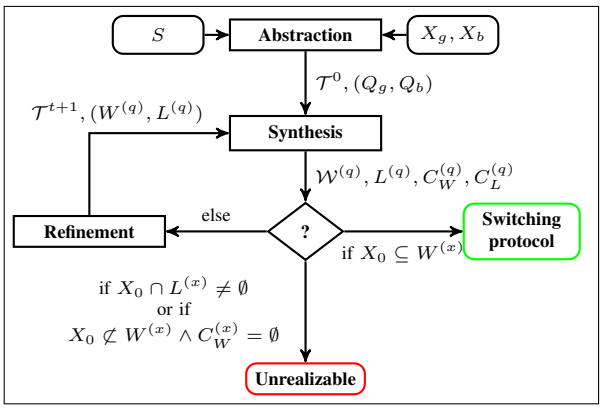

Fig. 1: Proposed abstraction refinement loop.

refining the abstraction may enable expansion of the losing set. The sets $W^{(q)}$, $L^{(q)}$ and $C_W^{(q)} \cup C_L^{(q)}$ are disjoint.

Strictly speaking, the winning set in the discrete reach-avoid-stay game is only $W^{(q)}$, and for all $q \in Q \setminus W^{(q)}$, non-determinism can prevent the system from safely reaching the goal set or from remaining within the goal set. The reason for computing $L^{(q)}$ is that it may provide useful information about the underlying continuous game which in turn can be used to determine unrealizability.

Let α be the abstraction function relating the states of the continuous system $\mathcal{S}$ to the states of the discrete system $\mathcal{T} \underset{\text{O.A.}}{\succeq} \mathcal{S}$. Then, by construction of $W^{(q)}$ and $L^{(q)}$, we have the following: (i) the set $W^{(x)} = \alpha^{-1}(W^{(q)})$ is a winning set for the continuous reach-avoid-stay game, (ii) the set $L^{(x)} = \alpha^{-1}(L^{(q)})$ contains only continuous states from which it is not possible to avoid the bad set X_b for all disturbances, (iii) the set $C_W^{(x)} = \alpha^{-1}(C_W^{(q)})$ contains continuous states from which there exist some trajectories entering $W^{(x)}$, hence these are candidate states to be part of the winning set, and, (iv) the set $C_L^{(x)} = \alpha^{-1}(C_L^{(q)})$ contains continuous states from which there exists at least one trajectory per mode entering $L^{(x)}$, hence these are candidate states to be part of the losing set.

If neither a switching protocol nor a certificate of unrealizability can be extracted from the synthesis at iteration k, the finite transition system $\mathcal{T}^t$ is refined to obtain $\mathcal{T}^{t+1}$ as explained in Section IV-D. After refinement, the synthesis method is called again with the goal set, bad set and initial state set derived from the winning set, losing set and the initial state set of the synthesis problem at the previous iteration via the abstraction function β from $\mathcal{T}^{t+1}$ to $\mathcal{T}^t$. From the properties of the refinement relations, it can be seen that the computed winning and losing sets of the switched system $\mathcal{S}$ are non-contracting through these iterations.

B. Abstraction

Given a switched system $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a\in\mathcal{A}}, D)$ together with an initial set X_0 and 'good' and 'bad' sets X_g and X_b, we first equip $\mathcal{S}$ with a set $\Pi = \{\pi_0, \pi_g, \pi_b\}$ of atomic propositions and an observation map h_X as in Corollary 1. Then, an initial over-approximation $\mathcal{T}^0 = (Q^0, \mathcal{A}, \rightarrow_{\mathcal{T}^0}, \Pi, h_{Q^0}, \mathcal{G}^0)$ is computed in two steps.

Firstly, the set of discrete states Q^0 is set in a way that there is a bijection between Q^0 and the coarsest partition P of X induced by the sets X_0, X_g and X_b. This uniquely defines an abstraction function $\alpha : X \to Q^0$, which satisfies for all $\xi_1, \xi_2 \in X$, $\alpha(\xi_1) = \alpha(\xi_2) \implies h_X(\xi_1) = h_X(\xi_2)$, and an observation map h_{Q^0} such that for all $\xi \in X$, $h_{Q^0}(\alpha(\xi)) = h_X(\xi)$. Secondly, the transitions $\rightarrow_{\mathcal{T}^0}$ and the progress group map $\mathcal{G}^0$ are computed using Algorithms 3 and 4 given in the Appendix.

C. Incremental synthesis algorithm

In this section, we present an incremental synthesis algorithm for reach-avoid-stay games $\langle \mathcal{T}, (Q_g, Q_b)\rangle$ on augmented finite transition systems. In addition to computing the winning and losing sets for the discrete reach-avoid-stay game, this algorithm computes two additional candidate sets $C_W^{(x)}$ and $C_L^{(x)}$ that provide potentially useful information about the underlying continuous reach-avoid-stay game. These sets are used as input to the refinement step.

The winning set of a reach-avoid-stay game can be characterized using a maximal controlled invariant set and a controllable predecessor set. Given an augmented finite transition system $\mathcal{T}$ and a set $Q_s \subseteq Q$, we give the following definitions:

Definition 4: The *maximal controlled invariant set* of $\mathcal{T}$ contained in Q_s, denoted $\mathbf{CInv}^\infty(Q_s)$, is the largest set of states in Q_s such that there exists a control strategy μ that guarantees that for any initial condition $q \in \mathbf{CInv}^\infty(Q_s)$, any μ-controlled state trajectory remains within $\mathbf{CInv}^\infty(Q_s)$.

Definition 5: The *controllable predecessor set* of Q_s in $\mathcal{T}$, denoted $\mathbf{CPre}^\infty(Q_s)$, is the set of all states in $\mathcal{T}$ from which there exists a control strategy μ that can enforce any μ-controlled state trajectory to visit Q_s.

These sets can be exactly computed for augmented finite transition systems using fixed point operations [20]. We also define the standard one step operators, which are used to find the candidate sets. The *guaranteed to reach* operator for action a is given by $\mathbf{Pre}_{a,\forall}(Q_s) \doteq \{q \in Q : \forall q' \ (q, a, q') \in \rightarrow_{\mathcal{T}}, q' \in Q_s\}$, and, the *no chance to avoid* operator is given by $\mathbf{Pre}_{\forall,\forall}(Q_s) \doteq \cap_{a\in\mathcal{A}}\mathbf{Pre}_{a,\forall}(Q_s)$. Similarly, *potential to reach* operators are given by $\mathbf{Pre}_{a,\exists}(Q_s) \doteq \{q \in Q : \exists q' \ (q, a, q') \in \rightarrow_{\mathcal{T}}, q' \in Q_s\}$, and, $\mathbf{Pre}_{\exists,\exists}(Q_s) \doteq \cup_{a\in\mathcal{A}}\mathbf{Pre}_{a,\exists}(Q_s)$. Finally, the *potential to not avoid* operator is given by $\mathbf{Pre}_{\forall,\exists}(Q_s) \doteq \cap_{a\in\mathcal{A}}\mathbf{Pre}_{a,\exists}(Q_s)$.

Algorithm 1 attempts to find a winning set given a discrete game $\langle \mathcal{T}, (Q_g, Q_b)\rangle$ and consists of three parts. Firstly, in order to satisfy the 'stay' part of the specification, a controlled invariant set inside of $Q_g \setminus Q_b$ needs to be found. If no such set can be found, the algorithm returns $C_W^{(q)} = Q_g$ for the goal set to be refined, hoping that a refined abstraction of X_g will reveal a controlled-invariant set.

Secondly, if a controlled invariant set W_0 was found inside of $Q_g \setminus Q_b$, this is a winning set. This set is then expanded using the first function of Algorithm 2 which returns the set W from which W_0 is reachable and a set C_W of states that has potential to transition to the winning set W. A refinement

in C_W may enable expansion of W at a later stage. Finally, the losing set is expanded to include states from where the current losing set is impossible to avoid, and a candidate losing set C_L is computed that can potentially expand the losing set if refined.

Roughly speaking, when a fixed-point operation is used while computing the winning or losing sets, candidate sets contain those states that are "neighbors" to the fixed-point.

Algorithm 1 Synthesis

1: **function** SYNTHESIS($\mathcal{T}$, Q_g, Q_b)
2: W_0 = **CInv**$^\infty(Q_g \setminus Q_b)$
3: **if** $W_0 \neq \emptyset$ **then**
4: $(W^{(q)}, C_W^{(q)})$ = EXPANDWINNING($\mathcal{T}, W_0, Q_b$)
5: **else**
6: $W^{(q)} = \emptyset, C_W^{(q)} = Q_g$
7: $(L^{(q)}, C_L^{(q)})$ = EXPANDLOSING($\mathcal{T}, Q_b$)
8: **return** $W^{(q)}$, $L^{(q)}$, $C_W^{(q)}$, $C_L^{(q)}$

Algorithm 2 Winning/losing set expansion

1: **function** EXPANDWINNING($\mathcal{T}$, W_0, Q_b)
2: $W = \mathbf{CPre}^\infty(W_0)$
3: $C_W = \mathbf{Pre}_{\exists,\exists}(W) \setminus (W \cup Q_b)$
4: **return** W, C_W
5: **function** EXPANDLOSING($\mathcal{T}$, L_0)
6: $\tilde{L} = L_0, L = \emptyset$
7: **while** $L \neq \tilde{L}$ **do**
8: $L = \tilde{L}$, $\tilde{L} = L \cup \mathbf{Pre}_{\forall,\forall}(L)$
9: $C_L = \mathbf{Pre}_{\forall,\exists}(L)$
10: **return** L, C_L

D. Refinement

Let $\mathcal{T}^t = (Q^t, \mathcal{A}, \rightarrow_{\mathcal{T}^t}, \Pi, h_{Q^t}, \mathcal{G}^t)$ with state space $Q^t = \{q_1, \dots q_k\}$ and an abstraction function $\alpha : X \rightarrow Q^t$. A refinement $\mathcal{T}^{t+1}$ of $\mathcal{T}^t$ is constructed by splitting a cell $\alpha^{-1}(q_i)$ corresponding to a state $q_i \in C_W^{(q)} \cup C_L^{(q)}$ into two parts, and assigning to each part a discrete state. The splitting policy can be chosen in different ways. For instance, the $\alpha^{-1}(q_i)$ with the largest volume could be split along some canonical middle line. The splitting creates two new discrete states q_{i_1} and q_{i_2} and thus a new set of states $Q^{t+1} = \{q_1, \dots, q_{i-1}, q_{i_1}, q_{i_2}, q_{i+1}, \dots, q_k\}$. We define the refined abstraction function $\alpha^{t+1} : X \rightarrow Q^{t+1}$ such that $\alpha^{t+1}(\xi) = \alpha^t(\xi)$ for $\xi \notin (\alpha^t)^{-1}(q_i)$, and such that $(\alpha^{t+1})^{-1}(q_{i_1})$, $(\alpha^{t+1})^{-1}(q_{i_2})$ form a 2-cell partition of $(\alpha^t)^{-1}(q_i)$. This allows to implicitly define an abstraction function $\beta : Q^{t+1} \rightarrow Q^t$ by $\alpha^t = \beta \circ \alpha^{t+1}$ and we can write $h_{Q^{t+1}} = h_{Q^t} \circ \beta$.

To compute transitions $\rightarrow_{\mathcal{T}^{t+1}}$ and a progress group map $\mathcal{G}^{t+1}$ for Q^{t+1}, Algorithms 3 and 4 in the Appendix could be run to create an augmented transition system from scratch. However, in order to mitigate the computational burden, $\rightarrow_{\mathcal{T}^{t+1}}$ can be constructed from $\rightarrow_{\mathcal{T}^t}$ using local modifications. To this effect, let $M : Q \times Q \rightarrow \{0, 1\}$ be a neighbor map of the state-space partition in the sense that $M(q, q') = 1$ if and only if $cl((\alpha^{t+1})^{-1}(q)) \cap cl((\alpha^{t+1})^{-1}(q')) \neq \emptyset$, where cl is set closure. Given M, $\rightarrow_{\mathcal{T}^{t+1}}$ is constructed as follows:

- For each $(q_j, a, q_k) \in \rightarrow_{\mathcal{T}^t}$ such that $i \notin \{j, k\}$, add (q_j, a, q_k) to $\rightarrow_{\mathcal{T}^{t+1}}$.
- For each action, run lines 5–6 of Algorithm 3 for all (q_j, q_k) such that $M(q_j, q_k) = 1$ and $\{j, k\} \cap \{i_1, i_2\} \neq \emptyset$.
- For each action, run lines 8–9 of Algorithm 3 for q_{i_1} and q_{i_2} and lines 10-11 for all q such that $M(q, q_{i_1}) = 1$ or $M(q, q_{i_2}) = 1$.

Similarly, the progress group map $\mathcal{G}^{t+1}(a)$ for action a can be updated as follows:

- For each $G \in \mathcal{G}^t(a)$ such that $q_i \in G$, add $(G \setminus \{q_i\}) \cup \{q_{i_1}, q_{i_2}\}$ to $\mathcal{G}^{t+1}(a)$.
- For each $G \in \mathcal{G}^t(a)$ such that $q_i \notin G$, if there exists a $q \in G$ with $M(q, q_{i_l}) = 1$ for some $l \in \{1, 2\}$ and $\cup_{q \in G}(\alpha^{t+1})^{-1}(q) \cup (\alpha^{t+1})^{-1}(q_{i_l})$ is transient on mode a, add $G \cup \{q_{i_l}\}$ to $\mathcal{G}^{t+1}(a)$; else add G to $\mathcal{G}^{t+1}(a)$.
- For $l \in \{1, 2\}$, if $(\alpha^{t+1})^{-1}(q_{i_l})$ is transient on mode a, add $\{q_{i_l}\}$ to $\mathcal{G}^{t+1}(a)$.

The following then holds by construction.

Proposition 2: Given a switched system $\mathcal{S}$ and an augmented finite transition system $\mathcal{T}^t$ such that $\mathcal{T}^t \underset{\text{O.A}}{\succeq} \mathcal{S}$, the result $\mathcal{T}^{t+1} = (Q^{t+1}, \mathcal{A}, \rightarrow_{T^{t+1}}, \Pi, h_{Q^{t+1}}, \mathcal{G}^{t+1})$ of the refinement procedure satisfies $\mathcal{T}^t \underset{\text{A.S.}}{\succeq} \mathcal{T}^{t+1} \underset{\text{O.A}}{\succeq} \mathcal{S}$.

E. Extraction of control law

The procedure presented in the previous subsections for computing a winning set can also be used to incrementally construct a winning strategy. In order to construct a winning strategy we extract a function $K : W^{(q)} \rightarrow 2^{\mathcal{A}}$ that maps a discrete winning state q to a (set of) "winning" mode(s) for that state. Every time a new state q is added to the winning set, a list of corresponding winning modes is appended to K using a procedure similar to Algorithm 1 in [20]. Given the list of winning modes K, a memoryless winning strategy $\mu : (q(i)) \mapsto a(i)$ can be constructed by ensuring that $a(i) \in K(q(i))$. This strategy μ, together with the abstraction function α, induces a winning switching protocol $\sigma : W^{(x)} \rightarrow \mathcal{A}$ for the underlying continuous reach-avoid-stay game in the form $\sigma(\xi) \doteq \mu(\alpha(\xi))$ for all $\xi \in W^{(x)}$.

V. COMPUTATIONAL ASPECTS

The algorithms outlined in the preceding sections are general in terms of dynamics and the abstraction function α. In practice, the applicability of the abstraction and refinement algorithms rely on how efficiently the subroutines for computing transitions (i.e., $isBlocked$) and progress groups (e.g., $isTransient$) can be implemented, which limits the choices of α and the feasible classes of dynamics. The properties of these subroutines are explained in the Appendix.

In particular, one natural restriction is to let the cells in the partition be defined by linear inequalities, so that the cells in the partition are polyhedra. In this case polynomial time algorithms exist for implementing $isBlocked$ and $isTransient$ for both linear and polynomial dynamics, using linear programming and sum-of-squares programming, respectively. For example, to evaluate $isBlocked(\mathcal{P}_1, \mathcal{P}_2, f_a, D)$ for two

closed polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, the objective $f_a \cdot \hat{n}$ is maximized over $\mathcal{P}_1 \cap \mathcal{P}_2$ (the common facet of $\mathcal{P}_1$ and $\mathcal{P}_2$ with normal direction $\hat{n}$ pointing towards $\mathcal{P}_2$). If the maximal value is less than zero, no flow exists.

To reduce the computational effort further, the partition of X can be constructed from hyper boxes $\{(x_1, \dots x_n) : x_i \in [x_i^-, x_i^+]\}$, which constitute a special case of polyhedra. One advantage of hyper boxes is that their representation has fixed complexity (linear in the state-space dimension). Another advantage is that the vertices of a hyper box are easy to enumerate. Since the optimal value of a linear program over a facet is attained in one of its vertices, it is enough to consider the sign of the objective function at each vertex in order to evaluate $isBlocked$. This makes hyper boxes in combination with linear dynamics easy to handle computationally for low state-space dimensions (e.g., $n \leq 5$). In the examples that follow we will work with hyper boxes.

VI. Examples

In this section we present two examples to demonstrate the effectiveness of the proposed approach. The first example compares finite transition systems with augmented finite transition systems when used within the proposed framework. The second example compares the proposed abstraction-refinement based synthesis with the approach in [20] that uses a uniform partition of the state-space.

A. Numerical example

We consider a continuous-time switched system with three modes $a \in \mathcal{A} \doteq \{1, 2, 3\}$ and with the following dynamics: $f_1 = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 \\ x_1 - x_2^2 + 2 \end{bmatrix}$, $f_2 = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 \\ x_1 - x_2 \end{bmatrix}$, $f_3 = \begin{bmatrix} -x_2 - 1.5x_1 - 0.5x_1^3 + 2 \\ x_1 + 10 \end{bmatrix}$. Disturbances are not considered, i.e., $D = \emptyset$. The domain, the bad set and the goal set are $X = [-2, 2] \times [-1.5, 3]$, $X_b = [-2, -1] \times [-1.5, -1] \cup [1, 2] \times [2.5, 3]$, $X_g = [-1, -0.5] \times [1.5, 2]$, respectively. We consider a reach avoid-game (ignoring the stay part) of the form $\langle \mathcal{S}, (X_g, X_b) \rangle$ and demonstrate the applicability of the proposed approach in computing winning sets of such games with polynomial dynamics. The upper part of Fig. 2 shows the result of the proposed method when computing a winning set by running the abstraction-refinement loop using 100 iterations. The computed winning set $W^{(x)}$ is the white area.

As a comparison, we also computed a winning set using the same abstraction-refinement loop and the same number of iterations, but this time with finite transition systems defined in [2] as overapproximations instead of augmented finite transition systems. The lower part of Fig. 2 shows the result of this experiment with the resulting winning set $\hat{W}^{(x)}$ depicted in white. As can be seen from the fact that $W^{(x)}$ spans a larger region than $\hat{W}^{(x)}$, it is possible to achieve better results using augmented finite transition systems within the proposed abstraction-refinement loop with the same number of iterations.

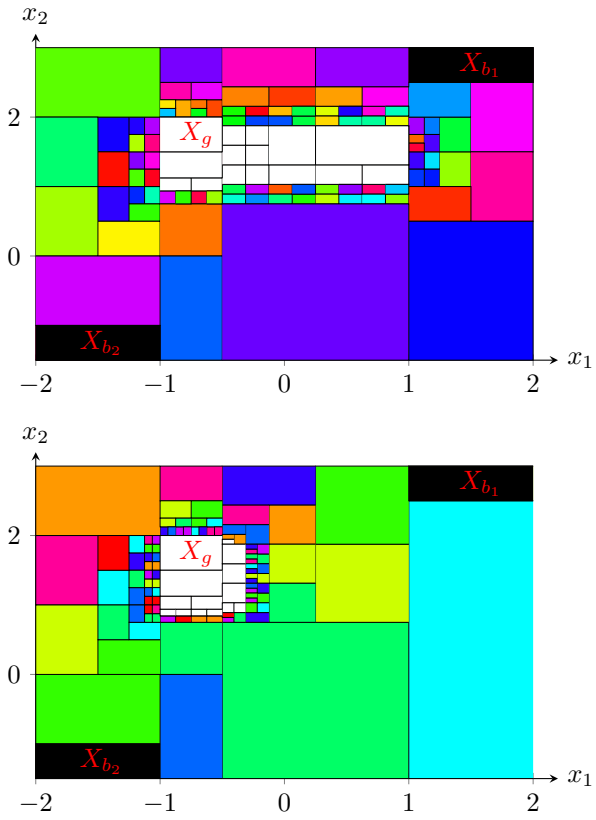

Fig. 2: Abstraction-refinement-based winning set computation when *augmented* finite transition systems (above) and finite transition systems (below) are used as overapproximations. X_g is the goal set, $X_{b_1} \cup X_{b_2}$ is the bad set, and white regions constitute the resulting winning sets $W^{(x)}$ (above) and $\hat{W}^{(x)}$ (below) on iteration K=100.

B. Radiant systems in buildings

In this section, we consider a hydronic radiant system for buildings. In the hydronic radiant system, hot or chilled supply water is pumped through the system tubes (i.e., the piping system) in order to adjust the temperature of the room.

We use the example from [20] where there are two zones equipped with one radiant system. The dynamics of this system can be modeled as a switched system with two modes and with states T_c (temperature of the pipe), T_1, T_2 (temperatures of zones). When the pump is circulating water in its piping system, Mode 0 dynamics are active:

$$C_r \dot{T}_c = \sum_{i=1}^{2} K_{r,i}(T_i - T_c) + K_w(T_w - T_c),$$
$$C_i \dot{T}_i = K_{r,i}(T_c - T_i) + K_i(T_a - T_i) + \sum_{j \neq i} K_{ij}(T_j - T_i) + q_i,$$

for $i = 1, 2$, where T_a is the ambient air temperature and T_w is the temperature of the supply water. Similarly, when the pump is not running, Mode 1 dynamics are active, which is identical to Mode 0 except that K_w is set to zero. For more details and parameter values for the system, see [24].

We set the domain as $20 \leq T_c, T_{1,2} \leq 28$. The objective is to control the zone temperatures to a desired range, denoted by a proposition SET ($21 \leq T_c \leq 27$ and $22 \leq T_{1,2} \leq 25$), and guarantee invariance of this range. The equilibrium

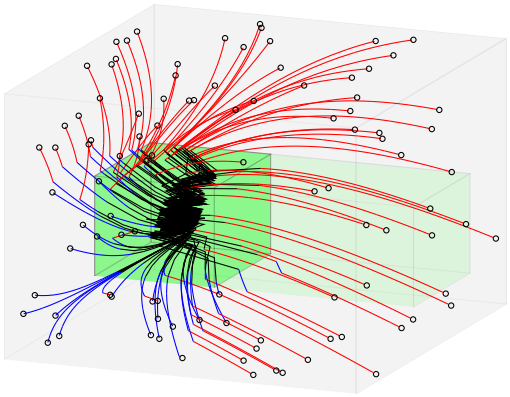

Fig. 3: Plot showing 100 sample trajectories of (2) over $t \in [0, 0.7]$ with randomized initial conditions in the winning set. The controlled invariant set is green, the goal set lighter green, and the winning set light gray. Red trajectory pieces correspond to f_1 and blue to f_2. Segments inside of the control-invariant set are plotted in black.

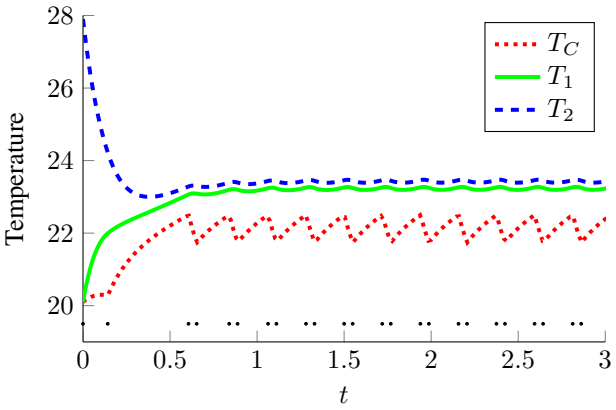

Fig. 4: Simulation of (2). Temperatures converge to the desired ranges. Black dots represent switches.

points of (2) are outside of SET, so the problem does not have a trivial solution. The abstraction-synthesis-refinement loop in Section IV on a partition based on hyper boxes is used with the following splitting policy:

- In the search for a controlled invariant set, split each hyper box in the goal set along its longest dimension,
- When trying to expand the winning set, split the largest hyper box in $C_W^{(x)} \cup C_L^{(x)}$ along its longest dimension.

In this example the volume of the domain is 512. After 7 iterations a controlled invariant set consisting of 12 hyper boxes is found, of total volume 20.25. A winning set consisting of 705 discrete states with total volume 424 is found after 53 iterations. In fact, this winning set is exactly the set ($20 \leq T_c \leq 26.625$ and $20 \leq T_{1,2} \leq 28$), shown in Fig. 3. The computation time was approximately 1h and 20 min on a 3.4 GHz iMac. In order to prevent excessive switching, the controlled invariant set has been further refined to 48 cells, and we use a control policy that keeps the control action constant when there are multiple choices (c.f Section IV-E). A plot of temperatures versus time for a simulation is shown in Figure 4.

As a comparison with the earlier work [20], our example used a slightly smaller target set SET. Yet we were able to find a larger winning set in terms of volume. Our winning set covered approximately 82% of the domain, while the corresponding number in the referred work was around 57%. Moreover, the overall computation took less than one third of the time compared to [20], although using a faster computer.

VII. Conclusions

In this paper, we presented a computational framework for synthesis of switching protocols. This framework uses augmented finite transition systems to abstract continuous-time switched systems. The main novelty of the presented approach is the integration of an abstraction refinement procedure with an incremental synthesis method. As shown in the examples, increasing the state-space size only as needed within the abstraction refinement loop provides computational advantages against approaches that work with uniform partitions of the state space. Another novelty of the presented approach is its ability to provide unrealizability certificates, which, for instance, can be used to guide the low-level control design in the case with switching between low-level feedback controllers.

In future work, we plan to investigate the trade-offs between the representations for partitions and efficiency of subroutines involved in our framework. Also, we will investigate parallel implementations of the abstraction refinement loop. Since the switching protocol synthesis problem is undecidable in general, the proposed abstraction-refinement loop is not guaranteed to terminate. Identifying classes of system for which termination can be guaranteed is another interesting direction. Finally, a thorough experimentation and comparison with the existing literature is needed.

Appendix

In this Appendix we outline the abstraction algorithms from [12] with some details of the subroutines relevant to abstraction-refinement. Given a switched system $\mathcal{S} = (X, \mathcal{A}, \{f_a\}_{a\in\mathcal{A}}, D, \Pi, h_X)$, a mapping $\alpha : X \to Q$ defining discrete states, and a neighbor map M as in Section IV-D, we compute transitions, observation map, and a progress group as follows:

- Define the observation map as $h_Q = h_X \circ \alpha^{-1}$.
- Compute $\mathcal{G} = \textsc{CompProgGroups}(\{f_a\}_{a\in\mathcal{A}}, D, Q, \alpha)$.
- Compute $\to_\mathcal{T} = \textsc{CompTrans}(\{f_a\}_{a\in\mathcal{A}}, D, Q, \alpha, M)$.
- Return $\mathcal{T} = (Q, \mathcal{A}, \to_\mathcal{T}, \Pi, h_Q, \mathcal{G})$.

The functions CompTrans and CompProgGroups are detailed in Algorithms 3 and 4. The intuition behind these functions are explained next. To find a transition relation $\to_\mathcal{T}$ that satisfies condition 2) of Definition 2, we must encode a transition (q_1, a, q_2) in $\to_\mathcal{T}$ whenever a trajectory of f_a starting in $\xi_1 \in \alpha^{-1}(q_1)$ and ending in $\xi_2 \in \alpha^{-1}(q_2)$ exists. Since the dynamics of the switched system are continuous in both state and time, it is enough to look for transitions on boundaries between cells, which is done by the subroutine

$isBlocked$. To find self-transitions (q, a, q) and progress groups the subroutine $isTransient$ is used to look for invariant sets. Lines 10-11 of Algorithm 3 are worth commenting on; they remove all transitions from a given state q under an action a, if there exists a continuous mode a-trajectory starting in q that exits X. This is required since exiting the continuous domain is forbidden. For the correctness of the algorithm we therefore require the following for the two subroutines:

R.1) $isBlocked(\mathcal{P}_1, \mathcal{P}_2, f, D)$: Given two sets $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{R}^n$, a vector field $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ and a disturbance set D, return $True$ if a certificate for the non-existence of a continuous trajectory segment (that does not go through a third set) from $\mathcal{P}_1$ to $\mathcal{P}_2$ under the flow of f is found, and return $False$ otherwise.

R.2) $isTransient(\mathcal{P}, f, D)$: Given a set $\mathcal{P} \in \mathbb{R}^n$, a vector field $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ and a disturbance set D, return $True$ if a certificate for transience of $\mathcal{P}$ under the flow of f is found, and return $False$ otherwise.

Depending on the dynamics and complexity of the sets in the given partition, it might not be always possible to efficiently to implement $isBlocked$ and $isTransient$. However, as shown in [12], even with conservative implementations of these subroutines, the abstraction procedure outputs an over-approximation.

Algorithm 3 Compute transitions $\to_{\mathcal{T}}$

1: **function** COMPTRANS($\{f_a\}_{a\in\mathcal{A}}$, D, Q, α, M)
2: initialize $\to_{\mathcal{T}} = \emptyset$
3: **for** $a \in \mathcal{A}$ **do**
4: **for** (q_i, q_j) such that $M(q_i, q_j) = 1$ **do**
5: **if not** $isBlocked(\alpha^{-1}(q_i), \alpha^{-1}(q_j), f_a, D)$ **then**
6: add (q_i, a, q_j) to $\to_{\mathcal{T}}$
7: **for** $q \in Q$ **do**
8: **if not** $isTransient(\alpha^{-1}(q), f_a, D)$ **then**
9: add (q, a, q) to $\to_{\mathcal{T}}$
10: **if not** $isBlocked(\alpha^{-1}(q), \mathbb{R}^n \setminus X, f_a, D)$ **then**
11: remove (q, a, q') from $\to_{\mathcal{T}}$ for all $q' \in Q$
12: **return** $\to_{\mathcal{T}}$

Algorithm 4 Compute progress group map $\mathcal{G}$

1: **function** COMPPROGGROUPS($\{f_a\}_{a\in\mathcal{A}}$, D, Q, α)
2: initialize $\mathcal{G} : \mathcal{A} \to 2^{2^Q}$
3: **for** $a \in \mathcal{A}$ **do**
4: initialize $\mathcal{G}(a) = \emptyset$
5: **for** $Q' \in 2^Q$ **do**
6: $\mathcal{X} = \cup_{q\in Q'} \alpha^{-1}(q)$
7: **if** $isTransient(\mathcal{X}, f_a, D)$ **then**
8: add $\mathcal{Q}'$ to $\mathcal{G}(a)$
9: **return** $\mathcal{G}$

ACKNOWLEDGMENT

This work was supported in part by NSF Contract #CNS-1239037 and the University of Michigan startup funds. The authors would like to thank Richard M. Murray of Caltech, Jun Liu of Univ. of Sheffield, and Pavithra Prabhakar of IMDEA for helpful discussions at early stages of this work.

REFERENCES

[1] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli, "Effective synthesis of switching controllers for linear systems," *Proc. IEEE*, vol. 88, no. 7, pp. 1011–1025, 2000.

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

[3] J. Hespanha and A. Morse, "Switching between stabilizing controllers," *Automatica*, vol. 38, no. 11, pp. 1905 – 1917, 2002.

[4] D. Liberzon and A. Morse, "Basic problems in stability and design of switched systems," *IEEE Control Systems Magazine*, vol. 19, no. 5, pp. 59–70, 1999.

[5] E. Frazzoli, M. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[6] H.-W. Park, A. Ramezani, and J. Grizzle, "A finite-state machine for accommodating unexpected large ground height variations in bipedal robot walking," *IEEE Trans. on Robotics*, vol. 28, no. 2, pp. 331 – 345, 2013.

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

[8] J. Ding and C. Tomlin, "Robust reach-avoid controller synthesis for switched nonlinear systems," in *Proc. of IEEE CDC*, 2010, pp. 6481–6486.

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

[10] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Trans. on Automatic Control*, vol. 57, no. 6, pp. 1491 –1504, 2012.

[11] E. Gol, X. Ding, M. Lazar, and C. Belta, "Finite bisimulations for switched linear systems," in *Proc. of IEEE CDC*, 2012, pp. 7632–7637.

[12] N. Ozay, J. Liu, P. Prabhakar, and R. Murray, "Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems," in *Proc. of American Control Conf.*, 2013.

[13] R. Alur, T. Dang, and F. Ivancic, "Counterexample-guided predicate abstraction of hybrid systems," *Theor. Comput. Sci.*, vol. 354, no. 2, pp. 250–271, 2006.

[14] E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald, "Verification of hybrid systems based on counterexample-guided abstraction refinement," in *Proc. of TACAS*, 2003, pp. 192–207.

[15] S. Shoham and O. Grumberg, "A game-based framework for CTL counterexamples and 3-valued abstraction-refinement," in *Proc. of CAV*, 2003, pp. 275–287.

[16] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. of IEEE CDC held jointly with CCC*, 2009, pp. 2222–2229.

[17] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental control synthesis in probabilistic environments with temporal logic constraints," in *Proc. of IEEE CDC*, 2012, pp. 7658–7663.

[18] S. Livingston, P. Prabhakar, A. Jose, and R. Murray, "Patching task-level robot controllers based on a local μ-calculus formula," in *Proc. of the IEEE ICRA*, 2013.

[19] B. Yordanov, J. Tumova, I. Cerni, J. Barnat, and C. Belta, "Formal analysis of piecewise affine systems through formula-guided refinement," *Automatica*, vol. 49, no. 1, pp. 261 – 266, 2013.

[20] F. Sun, N. Ozay, E. M. Wolff, J. Liu, and R. M. Murray, "Efficient control synthesis for augmented finite transition systems with an application to switching protocols," in *Proc. of American Control Conf.*, 2014.

[21] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" *J. of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[22] E. Grädel, W. Thomas, and T. Wilke, Eds., *Automata, Logics, and Infinite Games: A Guide to Current Research*, ser. Lecture Notes in Computer Science, vol. 2500. Springer, 2002.

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

[24] T. Nghiem, G. Pappas, and R. Mangharam, "Event-based green scheduling of radiant systems in buildings," in *Proc. of American Control Conf.*, 2013.