Temporal Logic Control of Discrete-Time Piecewise Affine Systems

Boyan Yordanov, *Member, IEEE*, Jana Tůmová, Ivana Černá, Jiří Barnat, and Calin Belta, *Senior Member, IEEE*

***Abstract*—We present a computational framework for automatic synthesis of a feedback control strategy for a discrete-time piecewise affine (PWA) system from a specification given as a linear temporal logic (LTL) formula over an arbitrary set of linear predicates in the system's state variables. Our approach consists of two main steps. First, by defining appropriate partitions for its state and input spaces, we construct a finite abstraction of the PWA system in the form of a control transition system. Second, by leveraging ideas and techniques from LTL model checking and Rabin games, we develop an algorithm to generate a control strategy for the finite abstraction. While provably correct and robust to state measurements and small perturbations in the applied inputs, the overall procedure is conservative and expensive. The proposed algorithms have been implemented as a software package and made available for download. Illustrative examples are included.**

***Index Terms*—Control design, discrete time systems, formal specifications, piecewise linear approximation.**

I. Introduction

TEMPORAL logics and model checking [1] are customarily used for specifying and verifying the correctness of digital circuits and computer programs. Due to their resemblance to natural language, expressivity, and existence of off-the-shelf algorithms for model checking, temporal logics have the potential to impact several other areas. Examples include analysis of systems with continuous dynamics [2], control of linear systems from temporal logic specifications [3], [4], task specification and controller synthesis in mobile robotics [5]–[7] and specification and analysis of qualitative behavior of genetic networks [8]–[10].

Manuscript received August 03, 2010; revised April 01, 2011, September 25, 2011 and November 02, 2011; accepted November 20, 2011. Date of publication December 14, 2011; date of current version May 23, 2012. This work was supported in part by Grants ARO W911NF-09-1-0088, NSF CNS-0834260, and AFOSR FA9550-09-1-020 at Boston University and in part by Grants GAP202/11/0312, GD102/09/H042, and LH11065 at Masaryk University. Recommended by Associate Editor S. Reveliotis.

B. Yordanov was with the Department of Mechanical Engineering, Boston University, Boston, MA 02215 USA. He is now with Microsoft Research, Cambridge CB3 0FB, U.K. (e-mail: yordanov@microsoft.com).

J. Tůmová was with the Division of Systems Engineering, Boston University, Boston, MA 02215 USA. She is now with the Department of Informatics, Masaryk University, Brno 60200, Czech Republic (e-mail: xtumova@fi.muni.cz).

I. Černá and J. Barnat are with Department of Informatics, Masaryk University, Brno 60200, Czech Republic (e-mail: cerna@fi.muni.cz; barnat@fi.muni.cz).

C. Belta is with the Department of Mechanical Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: cbelta@bu.edu).

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

Digital Object Identifier 10.1109/TAC.2011.2178328

In this paper, we focus on piecewise affine systems (PWA) that evolve along different discrete-time affine dynamics in different polytopic regions of the (continuous) state space. PWA systems are widely used as models in many areas. They can approximate nonlinear dynamics with arbitrary accuracy and are equivalent with several other classes of hybrid systems [11]. In addition, there exist efficient techniques for the identification of such models from experimental data, which include Bayesian methods, bounded-error procedures, clustering-based methods, mixed-integer programming, and algebraic geometric methods (see [12] for a review).

We consider the following problem: given a PWA system with polytopic control constraints, and a specification in the form of a linear temporal logic (LTL) formula over linear predicates in the system's state variables, find a set of initial states and a feedback control strategy, such that all trajectories of the closed-loop system originating in the initial set satisfy the formula. Our approach consists of two main steps. First, by partitioning the state and input spaces, we construct a finite abstraction of the PWA system in the form of a control transition system. Second, by leveraging ideas and techniques from LTL model-checking [1] and Rabin games [13], we develop an algorithm to generate a control strategy for the finite abstraction.

The main contributions of this work are the following. First, we develop a procedure based on polyhedral operations for the construction of finite abstractions of PWA systems, suitable for the synthesis of control strategies (control transition systems). Second, we show that the stuttering behavior inherent in such abstractions (i.e., self transitions at a state that can be taken infinitely but do not correspond to real trajectories of the concrete system), which is also related to the well known Zeno behavior, can be characterized and this additional information decreases the conservatism of the method. Third, we implement a procedure for LTL control of finite, nondeterministic systems by extending a Rabin game algorithm to explicitly consider information about stuttering, which leads to additional winning strategies. Finally, we seamlessly integrate our abstraction and control procedures into a unified computational framework, allowing fully automatic generation of control strategies for PWA systems from high-level, rich LTL specifications. The resulting feedback control strategies are robust in the sense that the state of the system does not need to be measured precisely but only with respect to certain thresholds and correctness is guaranteed even under small perturbations in the applied inputs. The framework was implemented in MATLAB as the software package CONPAS2 and is freely downloadable at http://hyness.bu.edu/software.

This paper can be seen in the context of literature focused on the construction of finite quotients of infinite systems (see [14] for an earlier review) and the abstraction-based control of such systems. We embed piecewise affine systems into transition systems inspired by [15] and [3] where the existence of equivalent (bisimulation) quotients and control strategies under the assumption of controllability for discrete-time linear systems is characterized. Algorithmic procedures for the control of continuous-time linear systems are given in [4] (and extended in [16]), where the constructed deterministic (nondeterministic) abstractions are not equivalent to the original system but instead capture only a restricted but controllable subset of its behavior. In this paper, we follow a similar approach but focus on the more general class of discrete-time PWA systems, which leads to different abstraction procedures based on polyhedral operations. An alternative abstraction technique based on quantifier elimination for real closed fields and theorem proving has been proposed in [17].

The problem of controlling finite systems from temporal logic specifications is equivalent to (digital circuit) synthesis and can be approached using automata-theoretic methods [18]. For synthesis problems, both specifications and environmental constraints are captured as a single temporal logic formula and the goal is to find a satisfying system, while in control applications the system is given and we seek a control strategy guaranteeing the satisfaction of the specification by the closed-loop system. Generating a control strategy for a finite, deterministic discrete event system (DES) from CTL* specifications is considered in [19], where LTL specifications are treated explicitly as a particular case (LTL is a subset of CTL*) and a solution based on Rabin games is proposed. We follow the same approach, which we review in Section III, but consider nondeterministic systems instead. The control of nondeterministic DESs has been previously considered in other settings such as for μ-calculus specifications [20] (LTL can be translated to μ-calculus) and an explicit implementation solving an LTL control problem for nondeterministic systems appears in [16], where specifications are restricted to LTL formulas generated by deterministic Büchi automata. By relaxing this restriction, in this paper we increase the expressivity of the specification language significantly.

To solve Rabin games, we implement an algorithm from [21] (see details in Section III) but extend it to deal with stuttering behavior, which leads to additional winning strategies. The concept of stuttering has been established previously [22] and related work has focused on determining if a specification is closed under stuttering [23], in which case it can be expressed in the LTL-X fragment (LTL without the next operator) [24] and less conservative stutter bisimulation quotients can be constructed [22]. Our approach is different, since it does not require any special structure from either the specification or the quotient. Instead, we characterize individual transitions as stuttering while constructing the abstraction and use this additional information during the solution of the Rabin game, which reduces the conservatism of the overall method but does not restrict the expressivity of the specification.

Besides the tool CONPAS2, which we develop in this paper, the MPT TOOLBOX [25] and the HYBRID TOOLBOX [26] for MATLAB can also be used to design piecewise affine control laws but neither can handle the richness of LTL specifications directly. The problem of controlling mixed logical dynamical (MLD) systems from LTL specifications has been considered in [27] and is related to this work, due to the equivalence between MLD and PWA systems [11]. Rather than relying on the construction of finite quotients as in this paper, the approach taken in [27] involves representing LTL specifications as mixed-integer linear constraints but a finite horizon assumption is imposed. Other temporal logic control tools include PESSOA [28], LTLCON [4], and its extension BÜCON [16]. PESSOA is capable of generating control strategies for linear, nonlinear, and switched systems from temporal logic specifications but abstractions based on the notions of approximate simulation and bisimulation are constructed. LTLCON addresses the control problem for linear systems but only deterministic abstractions are allowed, which leads to very conservative results. Its extension, BÜCON, relaxes this but restricts the specification language to the fragment of LTL generated by deterministic Büchi automata. In contrast, CONPAS2 constructs nondeterministic abstractions of piecewise affine systems and generates control strategies from full LTL specifications, while conservatism is further reduced by identifying stuttering behavior.

This paper extends recent results on formal analysis of PWA systems [29]–[31] to a control framework and is based on preliminary results presented in [32] and [33]. In [32] we computed (restrictive) deterministic or nondeterministic abstractions of PWA systems and used the procedures from LTLCON [4] or BÜCON [16], respectively, to solve the finite control problem. In [33], we developed a control procedure that allowed full LTL specifications and could handle information about stuttering. In this paper, by developing a complete control strategy for nondeterministic transition systems from full LTL specifications and by characterizing and dealing with stuttering phenomena we significantly 1) reduce the conservativeness of our previous approach and 2) increase the expressivity of the specification language.

The remainder of the paper is organized as follows. In Section II we provide preliminaries used throughout the paper. In Section III, we discuss the problem of controlling finite, nondeterministic transition systems from LTL specifications (LTL control) and review a solution based on Rabin games. The problem we consider is formulated in Section IV, where we also present an overview of our approach. In Section V, we define the control transition system and outline an algorithm for its computation. In Section VI, we use the results from Section III and Section V to formulate a solution to the main problem. In Section VII, we discuss a strategy for reducing the conservatism of the overall method by characterizing the stuttering behavior inherent in the construction of the control transition system. We outline the complexity associated with the proposed approach in Section VIII and describe its implementation and results from its application in Section IX. We conclude with final remarks in Section X.

II. PRELIMINARIES

In this section, we outline the notation and mathematical preliminaries used through the paper. Given a set Q, we use $|Q|$,

2^Q, Q^*, and Q^ω to denote its cardinality, powerset (the set of all subsets), and sets of nonempty finite and infinite sequences of elements from Q, respectively.

A full dimensional *polytope* $\mathcal{X}$ is defined as the convex hull of at least $N+1$ affinely independent points in $\mathbb{R}^N$. Given a polytope $\mathcal{X}$, we denote the set of *vertices* of $\mathcal{X}$ by $\mathcal{V}(\mathcal{X})$ and their convex hull as $\mathcal{X} = \text{hull}(\{v \in \mathcal{V}(\mathcal{X})\})$, which provides the V-representation of $\mathcal{X}$.

Alternatively, $\mathcal{X}$ can be described in H-representation as the intersection of at least $N+1$ closed half spaces $\mathcal{X} = \{x \in \mathbb{R}^N | h_i^T x \leq k_i, i = 1, \ldots, M\}$, where $M \geq N+1$ and, for all $i = 1, \ldots, M$, $h_i \in \mathbb{R}^N$, $k_i \in \mathbb{R}$. For notational convenience, through the rest of this paper we write the H-representation of a polytope as $\mathcal{X} = \{x \in \mathbb{R}^N | Hx \leq K\}$, where $H \in \mathbb{R}^{M \times N}, K \in \mathbb{R}^M$ and the vector inequality is considered component-wise. Algorithms for the translation between the V- and H- representations of a polytope exist [25], [34]. A *facet* of $\mathcal{X}$ is the intersection of the polytope with one of the supporting hyperplanes $h_i^T x = k_i$ for some $i = 1, \ldots, M$ from its H-representation. A polytope $\mathcal{X}$ without its facets is called an *open polytope* and we use $cl(\mathcal{X})$ to denote the closure of $\mathcal{X}$ (i.e., the union of $\mathcal{X}$ and its facets).

Definition 1: A nondeterministic transition system is a tuple $\mathcal{T} = (Q, \Sigma, \delta, O, o)$, where Q and Σ are (possibly infinite) sets of states and inputs, $\delta : Q \times \Sigma \to 2^Q$ is a (nondeterministic) transition map, O is a set of observations, and $o : Q \to O$ is an observation map.

A transition $\delta(q, \sigma) = Q'$ indicates that, while the system is in state q it can make a transition to any state $q' \in Q' \subseteq Q$ under input σ. We denote the set of inputs available at state $q \in Q$ by $\Sigma^q = \{\sigma \in \Sigma | \delta(q, \sigma) \neq \emptyset\}$. A transition $\delta(q, \sigma)$ is *deterministic* if $|\delta(q, \sigma)| = 1$ and the transition system $\mathcal{T}$ is deterministic if for all states $q \in Q$ and all inputs $\sigma \in \Sigma^q$, $\delta(q, \sigma)$ is deterministic. The transition system $\mathcal{T}$ is *finite* if both its set of states Q and set of inputs Σ are finite. $\mathcal{T}$ is *deadlock free* if, for every state $q \in Q$, $\Sigma^q \neq \emptyset$. In this work, we consider only deadlock free transition systems.

An *input word* of the system is defined as an infinite sequence $\sigma_0\sigma_1\sigma_2 \ldots \in \Sigma^\omega$. A *trajectory* of $\mathcal{T}$ produced by input word $\sigma_0\sigma_1\sigma_2 \ldots$ and originating at state $q_0 \in Q$ is an infinite sequence $q_0 q_1 q_2 \ldots$ with the property that $q_k \in Q$, and $q_{k+1} \in \delta(q_k, \sigma_k)$, for all $k \geq 1$. We denote the set of all trajectories of $\mathcal{T}$ originating at q by $\mathcal{T}(q)$ (similarly, we use $\mathcal{T}(Q') = \cup_{q' \in Q'} \mathcal{T}(q')$ to denote the set of all trajectories of $\mathcal{T}$ originating in $Q' \subseteq Q$). A trajectory $q_0 q_1 q_2 \ldots$ defines an *(output) word* $o(q_0)o(q_1)o(q_2)\ldots$. The set of all (output) words generated by the set of all trajectories originating at state $q \in Q$ is called the *language* of $\mathcal{T}$ originating at q and is denoted by $\mathcal{L}_\mathcal{T}(q)$ (similarly, we use $\mathcal{L}_\mathcal{T}(Q') = \cup_{q' \in Q'} \mathcal{L}_\mathcal{T}(q')$ to denote the language of $\mathcal{T}$ originating in $Q' \subseteq Q$).

For an arbitrary set of states $Q' \subseteq Q$ and set of inputs $\Sigma' \subseteq \Sigma$, we define the set of states $Post_\mathcal{T}(Q', \Sigma')$ that can be reached from Q' in one step by applying an input in Σ' as

$$Post_\mathcal{T}(Q', \Sigma') = \{q \in Q | \exists q' \in Q', \exists \sigma \in \Sigma', q \in \delta(q', \sigma)\}. \tag{1}$$

Definition 2: A (history dependent) *control function* $\Omega : Q^* \to \Sigma$ for transition system $\mathcal{T} = (Q, \Sigma, \delta, O, o)$ maps a finite, nonempty sequence of states to an input of $\mathcal{T}$. A control function Ω and a set of initial states $Q_0 \subseteq Q$ provide a *control strategy* for $\mathcal{T}$.

We denote a control strategy by (Q_0, Ω), the set of all trajectories of the closed loop system $\mathcal{T}$ under the control strategy by $\mathcal{T}(Q_0, \Omega)$, and the set of all words produced by the closed-loop $\mathcal{T}$ as $\mathcal{L}_\mathcal{T}(Q_0, \Omega)$. For any trajectory $q_0 q_1 q_2 \ldots \in \mathcal{T}(Q_0, \Omega)$ we have $q_0 \in Q_0$ and $q_{k+1} \in \delta(q_k, \sigma_k)$, where $\sigma_k = \Omega(q_0 \ldots q_k)$, for all $k \geq 0$.

To specify temporal logic properties of trajectories of transition systems (and PWA systems, as it will become clear later) we use linear temporal logic [1], [22]. Informally, LTL formulas are inductively defined over a set of observations O, by using the standard Boolean operators [e.g., $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction)] and temporal operators, which include $\bigcirc$ ("next"), U ("until"), $\Box$ ("always"), and $\Diamond$ ("eventually"). LTL formulas are interpreted over infinite words, as those generated by the transition system $\mathcal{T}$ from Def. 1.[1] We denote the language of infinite words that satisfy the formula ϕ by $\mathcal{L}_\phi$. We say that a trajectory $q_0 q_1 q_2 \ldots$ of $\mathcal{T}$ satisfies an LTL formula ϕ if and only if the corresponding word $o(q_0)o(q_1)o(q_2)\ldots$ satisfies ϕ. Transition system $\mathcal{T}$ satisfies ϕ from state $q \in Q$ if and only if all trajectories originating at q satisfy the formula (i.e., $\mathcal{L}_\mathcal{T}(q) \subseteq \mathcal{L}_\phi$), which can be verified using off-the-shelf *LTL model-checking* algorithms [1].

III. LTL Control

In this section, we consider the problem of controlling finite transition systems from LTL specifications (LTL control). We briefly review previous results from [4], [16], [19] and implement a solution by adapting the Rabin game-based approach from [19]. The LTL specification is first translated into a Rabin automaton [35], [36] followed by the solution of a Rabin game [21] to generate the control strategy. The material in this section serves as background for the extensions we introduce in Section VII to deal with stuttering behavior.

The LTL control problem can be formulated as a dual to the model checking problem:

Problem 1: Given a finite transition system $\mathcal{T} = (Q, \Sigma, \delta, O, o)$ and an LTL formula ϕ, find a control strategy (Q_0, Ω) such that all trajectories $\mathcal{T}(Q_0, \Omega)$ of the closed loop system satisfy ϕ (i.e., $\mathcal{L}_\mathcal{T}(Q_0, \Omega) \subseteq \mathcal{L}_\phi$).

When $\mathcal{T}$ is deterministic, the problem can be solved through model-checking-based techniques [4]. If $\mathcal{T}$ is nondeterministic, the problem becomes harder but can be treated as a game played on a finite graph and approached using automata-theoretic methods. In [16], we proposed a solution for the particular case when the LTL formula can be translated into a deterministic Büchi automaton which limits the applicability of the method (e.g., specifications such as $\Diamond\Box\phi$ for any LTL formula ϕ cannot be handled).

A Rabin game based solution to the problem of controlling finite, deterministic systems from LTL specifications has been proposed in [19]. The procedure involves the translation of the LTL specification into a (possibly nondeterministic) Büchi automaton, its subsequent determinization as a Rabin automaton

[1]More generally, the semantics of LTL formulas can be given over infinite words in the power set of the set of observations [1]

[35], the construction of the product automaton of the system and the specification, followed by the solution of a Rabin game on this product. This is also the approach we follow here to solve the LTL control problem for finite, nondeterministic systems (Problem 1). We translate the LTL specification directly into a Rabin automaton using LTL2DSTAR [36] (although an intermediate Büchi automaton is implicitly constructed by the tool) and implement the Rabin game algorithm from [21] which we will extend further in Section VII to deal with stuttering behavior. The resulting control strategy takes the form of a "feedback automaton," which reads the current state of $\mathcal{T}$ and produces the control input to be applied at that state. Through the rest of this section, we provide details of this finite control procedure as background for the extensions we introduce in Section VII.

A (nondeterministic) Rabin automaton is a tuple $\mathcal{R} = (S, S_0, O, \delta_{\mathcal{R}}, F)$, where S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, O is the input alphabet,[2] $\delta_{\mathcal{R}} : S \times O \to 2^S$ is a transition map, and $F = \{(G_1, B_1), \ldots, (G_n, B_n)\}$ is the acceptance condition. $\mathcal{R}$ is deterministic if $|S_0| = 1$ and $|\delta_{\mathcal{R}}(s, o)| \leq 1$ for all $s \in S$ and $o \in O$. The semantics of a Rabin automaton is defined over infinite input words. A run of $\mathcal{R}$ over a word $o_0o_1o_2 \ldots \in O^\omega$ is a sequence $\rho = s_0s_1s_2\ldots$, where $s_0 \in S_0$ and $s_{k+1} \in \delta_{\mathcal{R}}(s_k, o_k)$ for all $k \geq 1$. Let $\inf(\rho)$ denote the set of states that appear in the run ρ infinitely often. An input word is accepted by an automaton if some run over it is accepting. A run ρ is accepting if $\inf(\rho) \cap G_i \neq \emptyset \wedge \inf(\rho) \cap B_i = \emptyset$ for some $i \in \{1, \ldots, n\}$. We denote by $\mathcal{L}_{\mathcal{R}}$ the language (i.e., the set of all words) accepted by $\mathcal{R}$.

Given a finite transition system $\mathcal{T} = (Q, \Sigma, \delta, O, o)$ and an LTL formula ϕ over O, we translate ϕ into a deterministic Rabin automaton $\mathcal{R} = (S, S_0, O, \delta_{\mathcal{R}}, F)$ using LTL2DSTAR [36] where $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_\phi$ [35]. We construct the *product automaton* $\mathcal{P} = (S_{\mathcal{P}}, S_{\mathcal{P}0}, \Sigma, \delta_{\mathcal{P}}, F_{\mathcal{P}})$ of $\mathcal{T}$ and $\mathcal{R}$ where:

- $S_{\mathcal{P}} = Q \times S$ is the set of states;
- $S_{\mathcal{P}0} = Q \times S_0$ is the set of initial states;
- Σ is the input alphabet;
- $\delta_{\mathcal{P}} : S_{\mathcal{P}} \times \Sigma \to 2^{S_{\mathcal{P}}}$ is the transition map, where $\delta_{\mathcal{P}}((q, s), \sigma) = \{(q', s') \in S_{\mathcal{P}} | q' \in \delta(q, \sigma) \text{ and } s' = \delta_{\mathcal{R}}(s, o(q))\}$;
- $F_{\mathcal{P}} = \{(Q \times G_1, Q \times B_1), \ldots, (Q \times G_n, Q \times B_n)\}$ is the Rabin acceptance condition.

The product automaton is a nondeterministic Rabin automaton with the same input alphabet Σ as $\mathcal{T}$. Each accepting run $\rho_{\mathcal{P}} = (q_0, s_0)(q_1, s_1) \ldots$ of $\mathcal{P}$ can be projected into a trajectory $q_0q_1 \ldots$ of $\mathcal{T}$, such that the word $o(q_0)o(q_1) \ldots$ is accepted by $\mathcal{R}$ (i.e., satisfies ϕ) and vice versa [37]. This allows us to reduce Problem 1 to finding a control strategy[3] $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$ for $\mathcal{P}$, such that each run of the closed-loop $\mathcal{P}$ satisfies the Rabin acceptance condition $F_{\mathcal{P}}$. This problem can be viewed as a *Rabin game* played on the product automaton between two players—a protagonist and an adversary. A play is initiated in a state of the product automaton and proceeds according to the following rule: at each state, the protagonist chooses an input to be applied and the adversary determines the next state to be visited under this input (i.e., the adversary resolves nondeterministic transitions). A play produces an infinite sequence of states (i.e., a run) and it is won by the protagonist if the produced run satisfies the Rabin condition. A solution to the Rabin game is a control strategy: a control function determining moves of the protagonist and a set of initial states called winning region, such that each play under the strategy is won by the protagonist. Since winning strategies for Rabin games are memoryless [38], the control function is simply a map $\pi_{\mathcal{P}} : S_{\mathcal{P}} \to \Sigma$.

To solve Rabin games, we implement the recursive algorithm based on the construction of attractor sets from [21]. Given a set $S' \subseteq S_{\mathcal{P}}$, the *protagonist's attractor* is the set of states from which the protagonist can enforce a visit to S', while the *adversary's attractor* is the set of states from which the protagonist cannot prevent a visit to S'.

Definition 3: The protagonist's direct attractor of S', denoted by $\mathsf{A}^1_P(S')$, is the set of all states $s \in S_{\mathcal{P}}$, such that there exists an input σ satisfying $\delta_{\mathcal{P}}(s, \sigma) \subseteq S'$.

Definition 4: The adversary's direct attractor of S', denoted by $\mathsf{A}^1_S(S')$, is the set of all states $s \in S_{\mathcal{P}}$, such that there exists a state $s' \in \delta_{\mathcal{P}}(s, \sigma) \cap S'$ for each input $\sigma \in \Sigma^s$.

In other words, by applying input σ the protagonist can enforce a visit to S' from state $s \in \mathsf{A}^1_P(S')$, regardless of the adversary's following choice. Similarly, the adversary can enforce a visit to S' from state $s \in \mathsf{A}^1_S(S')$, regardless which input σ has been chosen by the protagonist. The protagonist's attractor $\mathsf{A}_P(S')$ can be computed iteratively via computation of converging sequence $\mathsf{A}^*_{P0}(S') \subseteq \mathsf{A}^*_{P1}(S') \subseteq \ldots$, where $\mathsf{A}^*_{P0}(S') = S'$ and $\mathsf{A}^*_{Pi+1}(S') = \mathsf{A}^1_P(\mathsf{A}^*_{Pi}(S'))$. Intuitively, $\mathsf{A}^*_{Pi}(S')$ is the set from which a visit to the set S' can be enforced by the protagonist in at most i steps. The adversary's attractor $\mathsf{A}_S(S')$ is computed analogously.

The following computation is then performed for each pair $(G, B) \in F_{\mathcal{P}}$:

1) The adversary's attractor $\mathsf{A}_S(B)$ is computed and removed from the game $\mathcal{P}$.
2) The protagonist's attractor $\mathsf{A}_P(G)$ is computed within the current game and marked as a part of the winning region. The rest of the game becomes the new game $\mathcal{P}' = \mathcal{P} \setminus \mathsf{A}_P(G)$.
3) Steps 1) and 2) are recursively applied to $\mathcal{P}'$ until a fixpoint $\mathcal{P}_{fix}$ is reached.
4) A sub-game $\mathcal{P}_{fix}$ with Rabin condition $F_{\mathcal{P}} \setminus \{(G, B)\}$ is then solved.

The winning region $W_{\mathcal{P}}$ is computed as $\mathsf{A}_P(S')$ where S' is the union of all winning regions parts from step 2). The control function $\pi_{\mathcal{P}}$ can be easily computed and the winning strategy $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$ is obtained by allowing initial states only (i.e., $W_{\mathcal{P}0} = W_{\mathcal{P}} \cap S_0$).

In order to complete the solution to Problem 1, we adapt $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$ as a control strategy (Q_0, Ω) for $\mathcal{T}$. Although the control function $\pi_{\mathcal{P}}$ was memoryless, Ω is history dependent and takes the form of a feedback control automaton $\mathcal{C} = (S, S_0, Q, \tau, \pi, \Sigma)$, where the set of states S and initial states S_0 are inherited from $\mathcal{R}$, the set of inputs Q is the set of

[2]as it will become clear later, we deliberately use the same symbol for observations of $\mathcal{T}$ and inputs of $\mathcal{R}$

[3]Control strategies for Rabin automata (such as $\mathcal{P}$) are defined by a set of initial states $W_{\mathcal{P}0}$ and a control function $\pi_{\mathcal{P}}$ as for transition systems (Def. 2). The behavior of the closed loop system is analogous.

states of $\mathcal{T}$, and the memory update function $\tau : S \times Q \to S$ and output function $\pi : S \times Q \to \Sigma$ are defined as

$$\tau(s,q)=\delta_{\mathcal{R}}\left(s,o(q)\right) \text{ if } (q,s) \in W_{\mathcal{P}}, \tau(s,q)=\perp \text{ otherwise}$$
$$\pi(s,q)=\pi_{\mathcal{P}}\left((q,s)\right) \text{ if } (q,s) \in W_{\mathcal{P}}, \pi(s,q)=\perp \text{ otherwise}$$

The set of initial states Q_0 of $\mathcal{T}$ is given by $\alpha(W_{\mathcal{P}0})$, where $\alpha : S_{\mathcal{P}} \to Q$ is the projection from states of $\mathcal{P}$ to Q. The control function Ω is given by $\mathcal{C}$ as follows: for a sequence $q_0 \ldots q_n$, $q_0 \in Q_0$, we have $\Omega(q_0 \ldots q_n) = \sigma$, where $\sigma = \pi(s_n, q_n)$, $s_{i+1} = \tau(s_i, q_i)$, and $q_{i+1} \in \delta(q_i, \pi(s_i, q_i))$, for all $i \in \{0, \ldots, n\}$. It is easy to see that the product automaton of $\mathcal{T}$ and $\mathcal{C}$ will have the same states as $\mathcal{P}$ but contains only transitions of $\mathcal{P}$ closed under $\pi_{\mathcal{P}}$. Then, all trajectories of the closed loop $\mathcal{T}(Q_0, \Omega)$ satisfy ϕ and therefore (Q_0, Ω) is the solution to Problem 1.

IV. Problem Formulation and Approach

In this section, we formulate the problem of controlling a PWA system from LTL specifications. Let $\mathcal{X}, \mathcal{X}_l$, $l \in L$ be open polytopes in $\mathbb{R}^N$, where L is a finite index set, such that $\mathcal{X}_{l_1} \bigcap \mathcal{X}_{l_2} = \emptyset$ for all $l_1, l_2 \in L, l_1 \neq l_2$ and $cl(\mathcal{X}) = \bigcup_{l \in L} cl(\mathcal{X}_l)$, where $cl(\mathcal{X}_l)$ denotes the closure of $\mathcal{X}_l$. A discrete-time piecewise affine (PWA) control system is defined as

$$x_{k+1} = A_l x_k + B_l u_k + c_l, x_k \in \mathcal{X}_l, u_k \in \mathcal{U} \quad (2)$$

where, at each time step $k = 1, 2, \ldots, x_k \in \mathbb{R}^N$ is the state of the system, u_k is the input restricted to a polytopic set $\mathcal{U} \subset \mathbb{R}^M$, and $A_l \in \mathbb{R}^{N \times N}$, $B_l \in \mathbb{R}^{N \times M}$, $c_l \in \mathbb{R}^N$ are the system parameters for mode $l \in L$.

At each time step k the exact state of the system ($x_k \in \mathcal{X}_l, l \in L$) is unknown but we can observe the current mode l. The semantics of system (2) are given over words in L^ω. Informally, a trajectory of the system produces a word by listing the index of the polytope visited at each step (e.g., trajectory $x_1 x_2 x_3 \ldots$ satisfying $x_1, x_2 \in \mathcal{X}_{l_1}$ and $x_3 \in \mathcal{X}_{l_2}$ for some $l_1, l_2 \in L$ will produce word $l_1 l_1 l_2 \ldots$). We assume that polytope $\mathcal{X}$ is an invariant for all trajectories of the system (in Section V-B we will show that polytopic control constraints guaranteeing this can be computed). Therefore, only infinite words are produced which can be checked against the satisfaction of an LTL formula over L (see Section II). We consider the following problem.

Problem 2: Given a PWA system (2) and an LTL formula ϕ over L, find a control strategy, such that all trajectories of the closed loop system satisfy ϕ.

In order to complete the formulation of Problem 2, we need to formalize the definitions of a control strategy for a PWA system (2) and the satisfaction of LTL formulas by trajectories of (2). We do this through an embedding into a transition system, for which both LTL satisfaction (Section II) and a control strategy (Def. 2) are clearly defined.

Definition 5 (Embedding Transition System): The embedding transition system $\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$ for system (2) is defined as follows:

- $Q_e = \bigcup_{l \in L} \mathcal{X}_l$;
- $\Sigma_e = \mathcal{U}$;

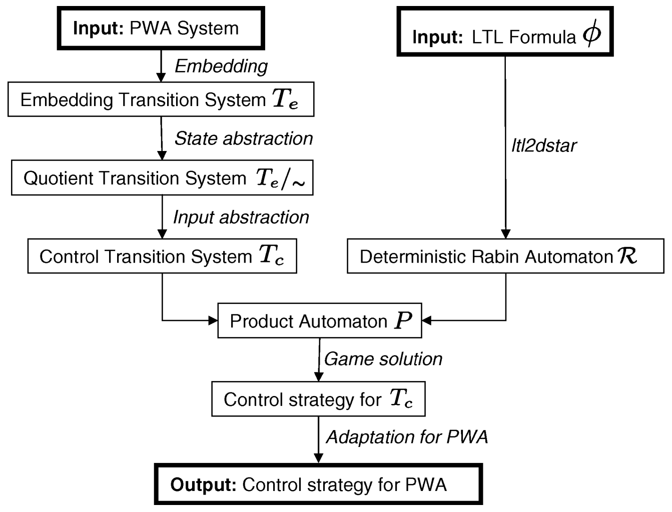
Input: PWA System
Embedding
Embedding Transition System T_e
State abstraction
Quotient Transition System $T_e/_\sim$
Input abstraction
Control Transition System T_c
Input: LTL Formula ϕ
ltl2dstar
Deterministic Rabin Automaton $\mathcal{R}$
Product Automaton P
Game solution
Control strategy for T_c
Adaptation for PWA
Output: Control strategy for PWA

Fig. 1. Illustration of our approach to Problem 2. First, we use the state equivalence relation induced by the polytopes from the definition of the PWA system to construct a quotient $\mathcal{T}_e/_\sim$, which has finitely many states but an infinite set of inputs. We then define an equivalence relation in the control space, which leads to the construction of the finite control transition system $\mathcal{T}_c$. This control transition system is synchronized with a deterministic Rabin automaton $\mathcal{R}$ that accepts the language satisfying the formula to produce a product automaton $\mathcal{P}$. A game-theoretic approach is then used to generate a memoryless control strategy for $\mathcal{P}$, which is translated to a (history dependent) control strategy for $\mathcal{T}_c$. The solution to Problem 2 is obtained by implementing the control strategy for T_c as a feedback control automaton for the initial PWA system that reads the index of the region visited at each step and supplies the next input.

- $\delta_e(x, u) = \{x'\}$ if and only if $x' \in Q_e$ and there exist $l \in L$ and $u \in \mathcal{U}$ such that $x \in \mathcal{X}_l$ and $x' = A_l x + B_l u + c_l$;
- $O_e = L$;
- $o_e(x) = l$ if and only if $x \in \mathcal{X}_l$.

Note that the embedding transition system $\mathcal{T}_e$ is always deterministic and non-blocking but both its set of states Q_e and set of inputs Σ_e are infinite.

Definition 6: Trajectories of system (2) originating in $Q_0 \subseteq Q_e$ satisfy formula ϕ if and only if $\mathcal{T}_e(Q_0)$ satisfies ϕ.

Problem 2 is an LTL control problem, where we seek a control strategy (Q_0, Ω) for the infinite, deterministic transition system $\mathcal{T}_e$. In Section III, we discussed the problem of controlling a finite, nondeterministic transition system from LTL specifications (Problem 1) and provided a solution based on Rabin games. Then, the overall approach to Problem 2 (illustrated in Fig. 1) involves the construction of a finite abstraction for $\mathcal{T}_e$ (referred to as the control transition system $\mathcal{T}_c$) such that a control strategy generated for $\mathcal{T}_c$ using the method from Section III can be adapted for $\mathcal{T}_e$. As it will become clear later, our approach is robust in the sense that the closed loop system is guaranteed to satisfy the specification even though absolute state measurements are not available and applied inputs are perturbed.

Remark 1: There are several simplifying assumptions in the formulation of Problem 2. First, we define the PWA system (2) on a set of open full-dimensional polytopes, thus ignoring states where the dynamics are ambiguous (states on the boundaries between regions), and capture the reachability of those sets only in the semantics of the embedding $\mathcal{T}_e$. This is enough for practical purposes, since only sets of measure zero are disregarded and it is unreasonable to assume that equality constraints can be detected in real-world applications. Trajectories starting and remaining in such sets are therefore of no interest. Trajectories starting in the interior of full-dimensional polytopes also cannot "vanish" in such zero measure sets unless the dynamics of the system satisfy some special conditions, which are easy to derive but omitted due to space constraints. Second, the specification is formulated over the polytopes $\mathcal{X}_l$, which are given *a priori*. However, arbitrary linear inequalities can be accommodated by including additional polytopes (as long as the region $l \in L$ visited at each step can be observed), in which case the system will have the same dynamics in several modes. Third, for notational simplicity we consider the mode (parameters) of system (2) to be determined only by the region of the state partition $\mathcal{X}_l, l \in L$ and not to depend on the applied input. Our methods can also handle a more general definition of PWA systems where the input space is also partitioned in a number of regions and different system dynamics for different combinations of state and input regions are defined.

V. Control Transition System

In this section, we define the control transition system $\mathcal{T}_c = (Q_c, \Sigma_c, \delta_c, O_c, o_c)$ for the embedding $\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$ (Def. 5) in Section V-A and present an algorithm for its computation in Section V-B. In Section VI, we will show how $\mathcal{T}_c$ is used to solve Problem 2.

A. Construction

The observation map o_e of $\mathcal{T}_e$ induces an equivalence relation $\sim$ over the set of states Q_e. We say that two states $x, x' \in Q_e$ are equivalent (written as $x \sim x'$) if and only if $o_e(x) = o_e(x')$. This equivalence relation induces a *quotient* transition system $\mathcal{T}_e/_\sim = (Q_e/_\sim, \Sigma_e, \delta_{e\sim}, O_e, o_{e\sim})$, where $Q_e/_\sim = L$ is the finite set of all equivalence classes formed in Q_e. The infinite set of inputs $\Sigma_e = \Sigma$ is preserved from $\mathcal{T}_e$ and the transitions of $\mathcal{T}_e/_\sim$ are defined as $l' \in \delta_{e\sim}(l, u)$ if and only if there exist $u \in \Sigma, x \in \mathcal{X}_l$ and $x' \in \mathcal{X}_{l'}$ such that $x' = \delta_e(x, u)$. Note that, in general, $\mathcal{T}_e/_\sim$ is nondeterministic, even though $\mathcal{T}_e$ is deterministic. Indeed, for a state of the quotient $l \in Q_e/_\sim$ it is possible that different states $x, x' \in \mathcal{X}_l$ have transitions in $\mathcal{T}_e$ to states from different equivalence classes under the same input. The set of observations $O_e = L$ of $\mathcal{T}_e/_\sim$ is preserved from $\mathcal{T}_e$ and the observation map $o_{e\sim}$ is identity. The transition map $\delta_{e\sim}$ can be related to the transitions of $\mathcal{T}_e$ by using the $Post$ operator defined in (1):

$$\delta_{e\sim}(l, u) = \{l' \in Q_e/_\sim | Post_{\mathcal{T}_e}(\mathcal{X}_l, \{u\}) \cap \mathcal{X}_{l'} \neq \emptyset\} \quad (3)$$

for all $l \in Q_e/_\sim$ and $u \in \Sigma_e$. For each state $l \in Q_e/_\sim$, we define an equivalence relation $\approx_l$ over the set of inputs Σ_e as $(u_1, u_2) \in \approx_l$ iff $\delta_{e\sim}(l, u_1) = \delta_{e\sim}(l, u_2)$. In other words, inputs u_1 and u_2 are equivalent at state l if they produce the same set of transitions in $\mathcal{T}_e/_\sim$. Let $U_l^{L'}$, $l \in L$, $L' \in 2^{Q_e/_\sim}$ denote the equivalence classes of Σ_e in the partition induced by the equivalence relation $\approx_l$:

$$U_l^{L'} = \{u \in \Sigma_e | \delta_{e\sim}(l, u) = L'\}. \quad (4)$$

Let $c(U_l^{L'})$ be an input in $U_l^{L'}$ such that $\forall u \in \Sigma_e, d(c(U_l^{L'}), u) < \epsilon \Rightarrow u \in U_l^{L'}$ where $d(u, u')$ denotes the distance between inputs $u, u' \in \Sigma_e$ and ϵ is a predefined parameter specifying the robustness of the control strategy. As it will become clear in Section V-B, $d(u, u')$ is the Euclidean distance in $\mathbb{R}^M$ and $c(U_l^{L'})$ can be computed as the center of a sphere inscribed in $U_l^{L'}$.

Initially, the states of $\mathcal{T}_c$ are the observations of $\mathcal{T}_e$ (i.e., $Q_c = L$). The set of inputs available at a state $l \in L$ is $\Sigma_c^l = \{c(U_l^{L'}) | L' \in 2^{Q_e/_\sim}\}$ and the transition map is $\delta_c(l, c(U_l^{L'})) = L'$. In general, it is possible that at a given state $l, \Sigma_c^l = \emptyset$, in which case state l is blocking. As it will become clear in Section V-B, such states are removed from the system in a recursive procedure together with their incoming transitions and therefore $Q_c \subseteq L$. The set of observations and observation map of $\mathcal{T}_c$ are preserved from $\mathcal{T}_e/_\sim$, which completes the construction of the control transition system.

Following from the construction described so far, the control transition system $\mathcal{T}_c$ is a finite transition system. In Section VI, we will show that a control strategy for $\mathcal{T}_c$ can be adapted as a robust control strategy (with respect to knowledge of exact state and applied input) for the infinite $\mathcal{T}_e$. This will allow us to use $\mathcal{T}_c$ as part of our solution to Problem 2.

B. Computation

Initially, the states of the control transition system $\mathcal{T}_c$ are simply the labels L of the polytopes from the definition of the PWA system (2). To complete its construction, we need to compute the set of inputs Σ_c^l available at each state $l \in Q_c$ and the transition map δ_c, while eliminating the states that are unreachable in order to guarantee that $\mathcal{T}_c$ remains deadlock free.

Given a polytope $\mathcal{X}_l$ from the definition of the PWA system (2), let

$$\Sigma^l = \{u \in \Sigma_e | Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \subseteq \mathcal{X}\} \quad (5)$$

be the set of all inputs guaranteeing that all states from $\mathcal{X}_l$ transit inside $\mathcal{X}$ (i.e., Σ^l is the set of all inputs allowed at l). In other words, regardless which $u \in \Sigma^l$ and $x \in \mathcal{X}_l$ are selected, x will transit inside $\mathcal{X}$ under u in $\mathcal{T}_e$. Then, in order to guarantee that $\mathcal{X}$ is an invariant for all trajectories of the system (an assumption that we made in the formulation of Problem 2) it is sufficient to restrict the set of inputs Σ_c^l available at each state $l \in Q_c$ to $\Sigma_c^l \subseteq \Sigma^l$.

Proposition 1: Let $\mathcal{X} = \{x \in \mathbb{R}^N | Hx < K\}$ be the H-representation of the polytope $\mathcal{X}$ from the definition of the PWA system (2). Then, Σ^l is a polytope with the following H-representation:

$$\Sigma^l = \{u \in \mathcal{U} | \forall v \in \mathcal{V}(\mathcal{X}_l), HB_l u < K - H(A_l v + c_l)\}. \quad (6)$$

Proof: See Appendix A. ■

The set of states reachable from state l in $\mathcal{T}_e/_\sim$ under the allowed inputs is

$$Post_{\mathcal{T}_e/_\sim}(l, \Sigma^l) = \{l' \in Q_e/_\sim | Post_{\mathcal{T}_e}(\mathcal{X}_l, \Sigma^l) \cap \mathcal{X}_{l'} \neq \emptyset\} \quad (7)$$

and can be computed using polyhedral operations, since[4]

$$Post_{\mathcal{T}_e}(\mathcal{X}_l, \Sigma^l) = A_l \mathcal{X}_l + B_l \Sigma^l + c_l. \tag{8}$$

Given a polytope $\mathcal{X}_l$ from the definition of the PWA system [(2)] and an arbitrary polytope $\mathcal{X}'$, let

$$U^{\mathcal{X}_l \to \mathcal{X}'} = \{u \in \Sigma_e | Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}' \neq \emptyset\} \tag{9}$$

denote the set of all inputs under which $\mathcal{T}_e$ can make a transition from a state in $\mathcal{X}_l$ to a state inside $\mathcal{X}'$. Equivalently, applying any input $u \in \mathcal{U}, u \notin U^{\mathcal{X}_l \to \mathcal{X}'}$ guarantees that $\mathcal{T}_e$ will not make a transition inside $\mathcal{X}'$, from any state in $\mathcal{X}_l$. The following proposition states that $U^{\mathcal{X}_l \to \mathcal{X}'}$ is a polytope that can be computed from the V- representations of $\mathcal{X}_l$ and $\mathcal{X}'$:

Proposition 2: Let H and K be the matrices in the H-representation of the following polytope:

$$\{\hat{x} \in \mathbb{R}^N | \exists x \in \mathcal{X}_l, A_l x + \hat{x} + c_l \in \mathcal{X}'\} \tag{10}$$

which can be computed from the V-representations of polytopes $\mathcal{X}_l$ and $\mathcal{X}'$ as

$$\text{hull}\{v' - (A_l v + c_l) | v \in \mathcal{V}(\mathcal{X}_l) v' \in \mathcal{V}(\mathcal{X}')\} \tag{11}$$

Then, $U^{\mathcal{X}_l \to \mathcal{X}'}$ is a polytope with the following H-representation:

$$U^{\mathcal{X}_l \to \mathcal{X}'} = \{u \in \mathcal{U} | H B_l u < K\} \tag{12}$$

Proof: See Appendix B. ■

Proposition 3: Given a state $l \in Q_c$ and a set of states $L' \in 2^{Q_c}$, the set $U_l^{L'}$ from (4) can be computed as follows:

$$U_l^{L'} = \bigcap_{l' \in L'} U^{\mathcal{X}_l \to \mathcal{X}_{l'}} \setminus \bigcup_{l'' \notin L'} U^{\mathcal{X}_l \to \mathcal{X}_{l''}}. \tag{13}$$

Proof: See Appendix C. ■

We can guarantee that if a state l' is not reachable from state l in $\mathcal{T}_e/_\sim$ (i.e., $l' \notin Post_{\mathcal{T}_e/_\sim}(l, \Sigma^l)$) then $U^{\mathcal{X}_l \to \mathcal{X}_{l'}} = \emptyset$ and therefore, $U_l^{L'} = \emptyset$ if $L' \not\subseteq Post_{\mathcal{T}_e/_\sim}(l, \Sigma^l)$; thus, the computation in (13) reduces to

$$U_l^{L'} = \bigcap_{l' \in L'} U^{\mathcal{X}_l \to \mathcal{X}_{l'}} \setminus \bigcup_{l'' \in Post_{\mathcal{T}_e/_\sim}(l, \Sigma^l) \setminus L'} U^{\mathcal{X}_l \to \mathcal{X}_{l''}}. \tag{14}$$

A non-empty input region $U_l^{L'}$ is in general nonconvex [see Eq. (14)] but can be represented as a finite union of open polytopes.[5] We define $r(U_l^{L'})$ and $c(U_l^{L'})$ as the radius and center of the largest sphere inscribed in a single polytope from $U_l^{L'}$. Note that in general this might be conservative (i.e., a sphere inscribed in a union of polytopes from $U_l^{L'}$ might have a larger radius) but finding a less conservative solution is beyond the scope of this paper.

In order to guarantee the robustness of the control strategy (as described in Section V-A) we only include input sets that are "large enough" (i.e., $r(U_l^{L'}) > \epsilon$, where ϵ is a predefined robustness parameter). In general, it is possible that blocking states appear in $\mathcal{T}_c$ (i.e., when $\Sigma_c^l = \emptyset$ for some l) and we make such states unreachable to guarantee that $\mathcal{T}_c$ is deadlock free. Following from the results presented in this section, the control transition system $\mathcal{T}_c$ can be computed using polyhedral operations only (the computation of $\mathcal{T}_c$ is summarized as Algorithm 1).

Algorithm 1 Construct $\mathcal{T}_c$

```
Input: PWA system [(2)] and robustness parameter $\epsilon$
Output: Control transition system $\mathcal{T}_c = (Q_c, \Sigma_c, \delta_c, O_c, o_c)$
1: $Q_c := L$
2: for each $l \in Q_c$ do
3:   $\Sigma_c^l := \Sigma^l$ [(5)]
4:   compute $Post_{\mathcal{T}_e/_\sim}(l, \Sigma_c^l)$ [(7)]
5:   for each $L' \subseteq Post_{\mathcal{T}_e/_\sim}(l, \Sigma_c^l)$ do
6:     compute $U_l^{L'}$ [(14)]
7:     if $r(U_l^{L'}) > \epsilon$ then
8:       include input $c(U_l^{L'})$ in $\Sigma_c^l$
9:       include transition $\delta_c(l, c(U_l^{L'})) = L'$
10:    end if
11:  end for
12: end for
13: $\Sigma_c = \bigcup_{l \in Q_c} \Sigma_c^l$
14: Make $\mathcal{T}_c$ deadlock free
15: return $\mathcal{T}_c$
```

VI. LTL Control of PWA Systems

In Section V, we defined the control transition system $\mathcal{T}_c$ as a finite abstraction of the infinite $\mathcal{T}_e$ and showed that it can be computed using polyhedral operations. In this section, we show that a control strategy generated for $\mathcal{T}_c$ using the approach from Section III can be adapted as a robust control strategy for the infinite $\mathcal{T}_e$. The satisfaction of LTL formulas by the closed-loop systems is preserved, which completes the solution to Problem 2.

Definition 7: A control strategy (Q_0^c, Ω^c) for $\mathcal{T}_c$ can be translated into a control strategy (Q_0, Ω) for $\mathcal{T}_e$ as follows. The initial set $Q_0^c \subseteq Q_c$ gives the initial set $Q_0 = \bigcup_{l \in Q_0^c} \mathcal{X}_l \subseteq Q_e$. Given a finite sequence of states $q_0 \ldots q_k$ where $q_0 \in Q_0$, the control function is defined as $\Omega(q_0 \ldots q_k) = \Omega^c(o_e(q_0) \ldots o_e(q_k))$.

The overall solution to Problem 2 consists of constructing the control transition system $\mathcal{T}_c$ (Section V), finding a satisfying control strategy (Q_0^c, Ω^c) for $\mathcal{T}_c$ (Section III) and adapting it as the control strategy (Q_0, Ω) for the original $\mathcal{T}_e$, or equivalently PWA system (Def. 7). At each step k, the input to be applied in $\mathcal{T}_c$ is given by the control function (i.e., $u_k^c = \Omega^c(q_0^c \ldots q_k^c)$) and we can guarantee that regardless which input $u_k \in \Sigma_e$ such that $d(u_k, u_k^c) < \epsilon$ is applied in $\mathcal{T}_e$, we have $\delta_e(q_k, u_k) \in \delta_c(q_k^c, u_k^c)$. This leads to the language inclusion $\mathcal{L}_{\mathcal{T}_e}(Q_0, \Omega) \subseteq \mathcal{L}_{\mathcal{T}_c}(Q_0^c, \Omega^c)$, which implies that if $\mathcal{T}_c(Q_0^c, \Omega^c)$ satisfies an arbitrary LTL formula ϕ, then so does $\mathcal{T}_e(Q_0, \Omega)$, and guarantees the correctness of the solution.

[4]Throughout this paper, when "+" connects sets it denotes their set (Minkowski) sum.

[5]Similarly to the situation discussed in Remark 1, a zero measure set is ignored in such a representation.

In order to implement a control strategy generated using this approach, at each step we only need to know which polytope $\mathcal{X}_l$ is visited by the PWA system but not the exact values of its state variables (i.e., we only need the observation l of a state of $\mathcal{T}_e$ rather than the state itself). Furthermore, perturbing the applied input does not affect the correctness of the control strategy, as long as the effective input is within the same input space sphere of radius ϵ, defined in Section V-B. This makes the generated control strategy robust to both state measurements and perturbations in the applied input—properties which can be tuned through the granularity of the state space partitioning of the PWA system or parameter ϵ, respectively. The more robust the required control strategy is, the more conservative the solution—a tradeoff we discuss in the following section, where we propose strategies for decreasing the conservatism of our method.

VII. Conservatism and Stuttering Behavior

In Section III, we described a solution to the problem of controlling a finite and possibly nondeterministic transition system from LTL specifications (Problem 1). In order to generate a control strategy for an infinite transition system such as $\mathcal{T}_e$ (Problem 2) we described the construction of a finite control abstraction $\mathcal{T}_c$ in Section V. However, due to *spurious trajectories* (i.e., trajectories of $\mathcal{T}_c$ not present in $\mathcal{T}_e$—a notion that will be explained in more detail through this section) we cannot guarantee that a control strategy will be found for $\mathcal{T}_c$ even if one exists for $\mathcal{T}_e$ and therefore, the overall method is conservative.

To reduce conservatism, in [30] we eliminated spurious trajectories through state refinement. Such a strategy can be adapted for our control procedure by either tessellating certain regions to refine the state partition or by using the dynamics of the system to find region subsets, from which trajectories can be controlled to other regions. However, any state refinement also requires more accurate measurement of the system's state, while in our problem formulation (Section IV) we assumed that the initial state partition captures certain physical limitations of our measurement capabilities (i.e., a control strategy cannot differentiate between states $x_1, x_2 \in Q_e$ when $o(x_1) = o(x_2)$). In the following, we focus on one particular source of conservatism (self loops at states of $\mathcal{T}_c$, which cannot be followed infinitely often), which is often present when abstractions are constructed, and develop a strategy for eliminating it.

A. Characterizing Stuttering Behavior

Inspired by the abstraction of *stutter* steps described in [22], in this paper we characterize only a specific class of spurious trajectories, which we introduce through an example (Fig. 2). Assume that a constant input $uuu\ldots$ produces a trajectory $x_1x_2x_3x_4\ldots$ in $\mathcal{T}_e$ where $o(x_1) = l_1, o(x_2) = o(x_3) = l_2, o(x_4) = l_3$ [Fig. 2(a)]. The corresponding word $l_1l_2l_2l_3\ldots$ is a trajectory of $\mathcal{T}_c$ (i.e., $l_1, l_2, l_3 \in Q_c$) and, from the construction described in Section V, it follows that $l_2 \in \delta_c(l_1, u)$ and $\{l_2, l_3\} \subseteq \delta_c(l_2, u)$ [Fig. 2(b)]. Then, there exists a trajectory of $\mathcal{T}_c$ that remains infinitely in state $l_2 \in Q_c$ under input u, which is not necessarily true for $\mathcal{T}_e$. Such spurious trajectories do not affect the correctness of a control strategy but increase

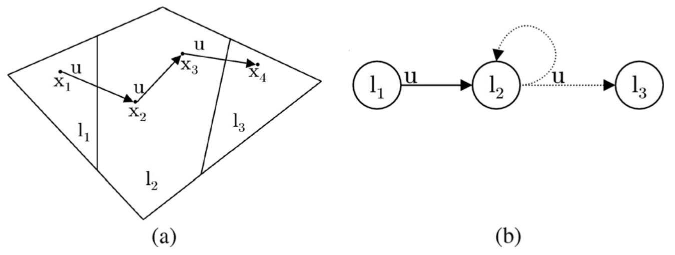

Fig. 2. Trajectory remaining forever in state l_2 exists in the finite abstraction (b), although such a behavior is not necessarily possible in the concrete system (a).

the overall conservativeness of the method. We address this by characterizing *stuttering inputs*, which guarantee that the system will leave a state eventually, rather than in a single step, and use this additional information during the construction of the control strategy for $\mathcal{T}_c$.

Definition 8: Given a state $l \in Q_c$ and a set of states $L' \in 2^{Q_c}$, the set of inputs $U_l^{L'}$ is *stuttering* if and only if $l \in L'$ and for all input words $u_0u_1\ldots$, where $u_i \in U_l^{L'}$, there exists a finite $k > 1$ such that the trajectory $x_0x_1\ldots$ produced in $\mathcal{T}_e$ by the input word satisfies $o(x_i) = l$ for $i = 1, \ldots, k-1$ and $o(x_k) = l' \in L', l' \neq l$.

Using Def. 8, we identify a stuttering subset $\Sigma_c^{ls} \subseteq \Sigma_c^l$ of the inputs available at a state $l \in Q_c$. Let $u = c(U_l^{L'}) \in \Sigma_c^l$ for some $L' \in 2^{Q_c}$ be an input of $\mathcal{T}_c$ computed as described in Section V. Then $u \in \Sigma_c^{ls}$ if and only if $U_l^{L'}$ is stuttering. Note that a transition $\delta_c(l, u) = L'$ from a state $l \in Q_c$ where u is stuttering is always nondeterministic (i.e., $|L'| > 1$) and contains a self loop (i.e., $l \in L'$) but the self loop cannot be taken infinitely in a row (i.e., a trajectory of $\mathcal{T}_e$ cannot remain infinitely in region $\mathcal{X}_l$ under input word $uuu\ldots$). An input $u \in \Sigma_c^{lu} = \Sigma_c^l \setminus \Sigma_c^{ls}$ induces a transition $\delta_c(l, u) = L'$ where: 1) when $L' = \{l\}$ trajectories of $\mathcal{T}_c$ and $\mathcal{T}_e$ produced by input word $uuu\ldots$ remain infinitely in state l and region $\mathcal{X}_l$, respectively, 2) when $l \notin L'$ trajectories of $\mathcal{T}_c$ and $\mathcal{T}_e$ leave state l and region $\mathcal{X}_l$, respectively, in one step under input u, 3) when $\{l\} \subset L'$ trajectories of $\mathcal{T}_e$ produced by input word $uuu\ldots$ can potentially remain in region $\mathcal{X}_l$ infinitely. Although in case 3) it is also possible that trajectories of $\mathcal{T}_e$ produced by input word $uuu\ldots$ leave region $\mathcal{X}_l$ in finite time, we have to be conservative in order to guarantee the correctness of the control strategy.

Note that our treatment of stuttering is different from [22] in two aspects. First, we require that $\mathcal{T}_c$ leaves a state after a finite number of transitions are taken under the same stuttering input and therefore an infinite sequence of stuttering self transitions is never possible. Second, we identify a set of stuttering inputs rather than constructing $\mathcal{T}_c$ as a time-abstract system. While we only characterize spurious infinite sequences of self loops (i.e., cycles of length 1), in general, it is possible that cycles of arbitrary length are spurious in $\mathcal{T}_c$. Considering higher order cycles is computationally challenging and decreases the conservativeness of the approach only for very specific cases, while spurious self loops are commonly produced during the construction of $\mathcal{T}_c$ and can be identified or constructed through polyhedral operations as described in Prop. 4 and 5.

Proposition 4: Given a state $l \in Q_c$ and a set of states $L' \in 2^{Q_c}$, input region $U_l^{L'}$ is stuttering if and only if $l \in L'$ and

$0 \notin \text{hull}\{(A_l-I)v_x+B_lv_u+c_l|\forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$, where I is the identity matrix.

Proof: See Appendix D. ■

Prop. 4 provides a computational characterization of stuttering input regions. In general, however, it is possible that an input region $U_l^{L'}$ cannot be identified as stuttering but a stuttering subset $\hat{U}_l^{L'} \subset U_l^{L'}$ can be identified. Then, if such a subset is "large enough" (i.e., $r(\hat{U}_l^{L'}) > \epsilon$) it can be used in $\mathcal{T}_c$ and allow more general control strategies. In Prop. 5 we describe the computation of such stuttering subsets.

Proposition 5: Given an arbitrary $a \in \mathbb{R}^N$, the input region $\hat{U}_l^{L'} = \{u \in U_l^{L'} | \forall v \in \mathcal{V}(\mathcal{X}_l), a^TB_lu > -a^T((A_l - I_N)v - c_l)\}$, where $l \in L'$ is always stuttering.

Proof: See Appendix E. ■

Although Prop. 5 is valid for an arbitrary $a \in \mathbb{R}^N$, the volume of the stuttering subset $\hat{U}_l^{L'} \subset U_l^{L'}$ depends on a. Since only "large enough" input regions are considered in $\mathcal{T}_c$ (see Alg. 1), a should be chosen in such a way that $r(\hat{U}_l^{L'})$ is maximized. This problem is beyond the scope of this paper but a possible (suboptimal) solution involves the uniform sampling of rotation groups as discussed in [39].

B. Rabin Games With Stuttering

The algorithm from Section III can be adapted to handle the additional information about stuttering inputs captured in $\mathcal{T}_c$, while the correctness and completeness of the control strategy computation for the product automaton $\mathcal{P}$ is still guaranteed. $\mathcal{P}$ is constructed as in Section III and therefore it naturally inherits the partitioned input set $\Sigma_c^l = \Sigma_c^{ls} \cup \Sigma_c^{lu}$ for each state $l \in Q_c$. Going back to the Rabin game interpretation of the control problem discussed in Section III, we need to account for the fact that a transition under the same stuttering input cannot be taken infinitely many times in a row. As a result, the construction of the control strategy is still performed using the algorithm from [21] and only the computation of the protagonist's direct attractor [Def. (3)] is modified as follows.

Let $l \in Q_c$ be a state and $u \in \Sigma_c^{ls}$ be a stuttering input of $\mathcal{T}_c$ (Def. 8). We are interested in *edge* (s, u, s') of transition $\delta_{\mathcal{P}}(s, u) = S'$, where $\alpha(s) = l$ and $s' \in S'$ (recall from Section III that $\alpha()$ is the projection from states of $\mathcal{P}$ to Q_c). Edge (s, u, s') is *u-nontransient* if $\alpha(s') = l$ and *transient* otherwise. A sequence of edges $(s_1, u_1, s_2)(s_2, u_2, s_3)\ldots(s_n, u_n, s_1)$, where $s_i \neq s_j$ for any $i, j \in \{1, \ldots, n\}$ is called a *cycle* and a cycle consisting of u-nontransient transitions only is called a *u-nontransient cycle*. The self loop (l, u, l) at state l under the stuttering input u cannot be followed infinitely many times in a row in $\mathcal{T}_c$. Consequently, in the product graph $\mathcal{P}$, a u-nontransient cycle cannot be followed infinitely many times in a row without leaving it, which leads to the following modification of Def. 3.

Definition 9: The protagonist's direct attractor of S', denoted by $\mathsf{A}_{\mathcal{P}}^1(S')$, is the set of all states $s \in S_{\mathcal{P}}$, such that there exists an input u where 1) $\delta_{\mathcal{P}}(s, u) \subseteq S'$ or 2) s lies on a u-nontransient cycle and, for all states s' of the cycle and all transient edges (s', u, s'') originating there, $s'' \in S'$.

In other words, the protagonist can enforce a visit to S' by following a u-nontransient cycle finitely many times and eventually leaving it to a state from S'. Even so, when applying input u the protagonist cannot prevent a visit to a state reachable via a u-nontransient edge but the adversary cannot enforce such a visit either (i.e., once input u is applied, following or avoiding a u-nontransient edge cannot be enforced by the protagonist or the adversary). In Section III, the adversary's attractor of S' could be interpreted equivalently as the set of states from which 1) the adversary can enforce a visit to S' or 2) the protagonist cannot prevent a visit to S'. When stuttering inputs are considered interpretation 1) is no longer consistent with Def. 4, whereas 2) still is. However, the Rabin game algorithm presented in Section III is based on interpretation 2) and, as a result, the definition of the adversary's direct attractor (Def. 4) remains unchanged. The protagonist's and adversary's attractors are computed from their respective direct attractors (Defs. 9 and 4) as in Section III.

By identifying stuttering inputs during the construction of the control transition system $\mathcal{T}_c$ (Props. 4 and 5) and modifying the approach from Section III through Def. 9 to handle this additional information during the construction of a control strategy for $\mathcal{T}_c$, we can reduce the conservatism associated with the overall method. Even so, our solution to Problem 2 remains conservative but it is important to note that the only source of conservativeness is the construction of $\mathcal{T}_c$—the solution to the LTL control problem for $\mathcal{T}_c$ is complete.

VIII. Complexity

As our proposed solution to Problem 2 consists of 1) the construction of the control transition system $\mathcal{T}_c$ and 2) the generation of a control strategy for $\mathcal{T}_c$, the overall computational complexity is the cumulative complexity of the two parts. The computation of $\mathcal{T}_c$ involves enumerating all subsets of L at any element of L, which gives $O(|L| \cdot 2^{|L|})$ iterations in the worst case, although in practice this can be reduced by considering only reachable states [see Eq. (14)]. At each iteration, polyhedral operations which scale exponentially with N (the size of the continuous state space) are performed [25], [34]. The characterization of stuttering inputs checks each element from Σ_c through polyhedral operations, which scale exponentially in M (the size of the input space). To simplify the analysis, we assume that $M = N$ (although in practice $M < N$) but the high theoretical complexity of translating between V- and H- representations of polytopes in higher dimensions [34] leads to an overall worst case complexity of $O(|L| \cdot 2^{|L|+N^2})$. In general, generating a control strategy is the more expensive part of our procedure and is discussed next. This computation depends on the number of states $|L|$ and transitions $|\Sigma_c|$ of $\mathcal{T}_c$ where, in the worst case, $|\Sigma_c| = |L| \cdot 2^{|L|}$ (although this is significantly reduced through the optimizations described later in this section).

While the finite abstraction $\mathcal{T}_c$ has at most $|L|$ states, the translation of an LTL formula ϕ into a Rabin automaton $\mathcal{R}$ leads to a doubly-exponential blowup [35], [37] and, therefore, the product automata $\mathcal{P} = \mathcal{T}_c \times \mathcal{R}$ has, in general, $n = |L| \cdot 2^{2^{\mathcal{O}(|\phi| \cdot \log|\phi|)}}$ states and $k = 2^{\mathcal{O}(|\phi|)}$ pairs in its acceptance condition (in practice, k is usually small as in the case studies summarized in Table I). When the algorithm from [21] is used to solve Rabin games, the complexity of the control strategy synthesis procedure described in Section III is $O(mn^{2k}k!)$, where m is the number of transitions in $\mathcal{P}$ and depends on the number of transitions $|\Sigma_c|$ of $\mathcal{T}_c$. Extending the algorithm to deal with

TABLE I
CASE STUDY COMPUTATION SUMMARY

#	1	2	3
$\|L\|$	49	36	36
$\|\Sigma_c\|_{\not\epsilon}$	242	949	949
$\|\Sigma_c\|$	146 (205)	222 (684)	222 (684)
$\|\Sigma_c^s\|$	59 (86)	109 (379)	109 (379)
n	196	108	180
m	584 (820)	666 (2052)	1105 (3420)
t_1	31sec (35sec)	67sec (59sec)	69sec (77sec)
t_2	41sec (92sec)	40sec (5min)	101sec (18min)

The number of states, input regions, large enough input regions, and stuttering inputs of the control transition system for each case study are denoted by $|L|$, $|\Sigma_c|_{\not\epsilon}$, $|\Sigma_c|$, and $|\Sigma_c^s|$, respectively. The number of states and transitions in the product automaton are denoted by n and m and, for all case-studies, there is only one acceptance pair ($k = 1$). Results for the computation of the control transition system (t_1) and the generation of the control strategy (t_2) for each case-study is given as a separate row, where results obtained when the optimization proposed in Section VIII is disabled are given in parentheses.

stuttering behavior as described in Section VII does not change the overall complexity. Computation times for the examples from Section IX, obtained on a 3.4-GHz, Intel Pentium 4 machine with 1 GB of memory, are summarized in Table I.

The doubly exponential blowup incurred from the translation of LTL into Rabin automata and the complexity of the game solution make the method computationally expensive. However, there are several strategies for reducing this complexity. First, it is possible to reduce the number of transitions of $\mathcal{T}_c$ after it is initially constructed without sacrificing solutions, which leads to reducing the number of transitions m in the product automaton. More "nondeterminism" available at a state does not result in more winning strategies for the algorithm described in Section III, while at the same time unnecessarily increases the complexity of the method. Formally, let $u_1 = c(U_l^{L_1'})$ and $u_2 = c(U_l^{L_2'})$ where $L_1', L_2' \in 2_c^Q, L_1' \subseteq L_2'$ be inputs of $\mathcal{T}_c$ available at state $l \in Q_c$ (i.e., $\{u_1, u_2\} \subseteq \Sigma_c^l$). If input u_2 is used in a control strategy, then the specification is satisfied regardless of which state $l' \in L_2'$ is visited in the next step. Clearly, the same holds for input u_1 since L_1' is a subset of L_2' but keeping both inputs is unnecessary. Therefore, at each state $l \in Q_c$ we set $\Sigma_c^{ls} = \Sigma_c^{ls} \setminus u_2$ if $u_1, u_2 \in \Sigma_c^{ls}$ or $\Sigma_c^{lu} = \Sigma_c^{lu} \setminus u_2$ if $u_1, u_2 \in \Sigma_c^{lu}$ when the property described above holds. Second, more efficient symbolic approaches can be implemented for solving Rabin games [40]. Third, better complexity can be achieved by restricting the specification language. By considering only specifications generated by deterministic Büchi automata, a single exponential formula translation is possible (this was the approach followed in [16]) while polynomial time algorithms solving games for specific LTL fragments are also available [41].

IX. Implementation and Case Studies

The method described in this paper was implemented in MATLAB as the software package *conPAS2*, where all polyhedral operations were performed using the MPT toolbox [25]. The tool takes as input a PWA system [as defined in Eq. (2)] and an LTL formula and produces a set of satisfying initial regions and a feedback control strategy for the system (see Section VI). The tool, including all described examples, is freely downloadable from our web site at http://hyness.bu.edu/software. To illustrate our methods, through the rest of this section we consider the problems of controlling a two-tank system and a synthetic gene network from LTL specifications.

First, we consider the two-tank system shown in Fig. 3(a). The system has two state variables ($N = 2$) that represent the water levels in the two tanks and range in (0,0.7). It has one control dimension ($M = 1$) representing the inflow rate, which ranges in $(0, 5e^{-4})$. The state space of the system is partitioned into 49 rectangular regions (i.e., $L = 1, \ldots, 49$) by 7 evenly spaced thresholds along each dimension. These thresholds signify that we can only detect whether the water level in each tank is above or below the marks at 0.1, 0.2, ..., 0.7 [see Fig. 3(b)]. Valve v_1 is opened only if submerged (i.e., if the water level in either tank is above 0.2—the height of the valve) and, therefore, the valve is closed when the system is in regions 1,2,8, and 9 and opened otherwise [see Fig. 3(a), (b)]. Discrete-time, linear equations describing the dynamics of the system in each mode are derived as in [42] using a 5-s. time step and $1.54e^{-2}$ m^2, $1e^{-4}$ m^2, and $2.125e^{-5}$ m^2 as the cross sectional areas of the two tanks, valve v_1, and valve v_2, respectively. The dynamics of the system for each region $l \in L$ are given by

$$A_l = \begin{cases} \begin{bmatrix} 1 & 0 \\ 0 & 0.9635 \end{bmatrix}, & \text{for } l = 1, 2, 8, 9 \\ \begin{bmatrix} 0.8281 & 0.1719 \\ 0.1719 & 0.7916 \end{bmatrix}, & \text{otherwise,} \end{cases}$$
$$B_l = [324.6753, 0]^T, \quad c_l = [0, 0]^T \quad \text{for} \quad l = 1, \ldots, 49$$

We seek a control strategy for the system guaranteeing the satisfaction of a specification, expressed informally as "whenever tank 2 is empty, it will eventually get filled up." To formalize this specification we define propositions π_1 := "the level of tank 2 is below 0.1" (i.e., "tank 2 is empty") and π_2 := "the level of tank 2 is above 0.4" (i.e., "tank 2 is full"), which can be expressed as disjunctions of regions from L as $\pi_1 := 1 \vee \ldots \vee 7$ and $\pi_2 := 29 \vee \ldots \vee 49$. We consider specification

$$\phi_1 = \Box(\pi_1 \Rightarrow \Diamond\pi_2). \tag{15}$$

Satisfying control strategies were found from all regions expect 49 when the required robustness was set to $\epsilon = 5e^{-6}$ [Fig. 3(b)]. If stuttering inputs are not characterized as described in Section VII, satisfying control strategies are identified for regions 29, ..., 48 only. A simulated trajectory of the closed-loop system is shown in Fig. 3(c) and computation times are given as case study #1 in Table I.

Besides the two-tank system discussed above, we apply our method to generate control strategies for a synthetic gene network inspired by the genetic toggle switch [43] [Fig. 4(a)]. The system has two state variables ($N = 2$) ranging in (0,100) that represent the concentrations of the two proteins and the state space is partitioned into 36 rectangular regions (i.e., $L = 1, \ldots, 36$). The system has two control dimension ($M = 2$) representing external control over the expression rates from the two promoters and ranging in (−15, 15) and (−18, 18). Gene regulation is captured by piecewise affine ramp functions which induce three ranges of different dynamics.

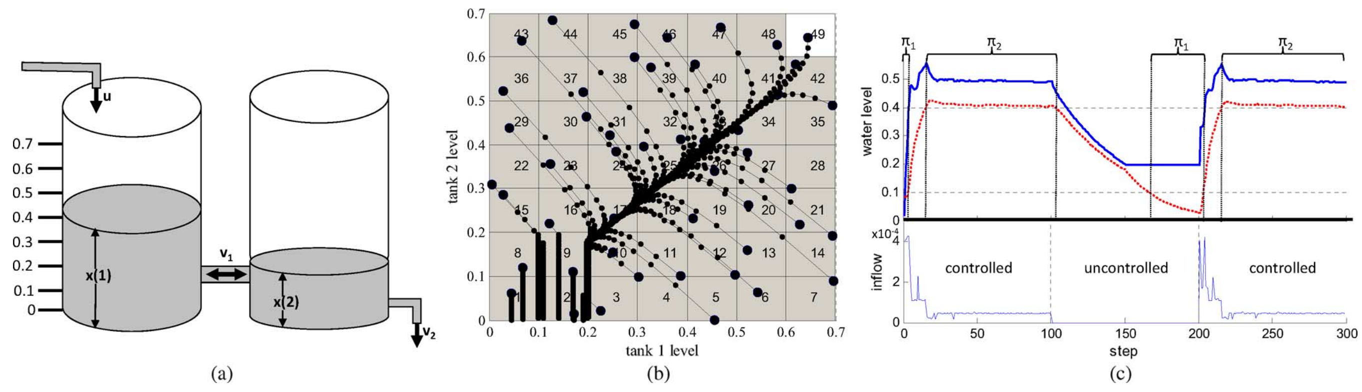

Fig. 3. (a) A two-tank system. Water is drained at a constant rate from tank 2 though valve v_2, while tank 1 is filled at a rate that is controlled externally. Water can also flow in either direction, from the tank with more water to the one with less, through valve v_1, which is opened only if submerged. (b) Simulated trajectories of the uncontrolled system. Initial conditions are shown as blue circles. Control strategies guaranteeing the satisfaction of specification $\phi_1 = \Box(\pi_1 \Rightarrow \Diamond\pi_2)$ are found from all shaded regions. (c) A simulated trajectory of the closed loop system. The water levels of tanks 1 and 2 are shown respectively as a blue (solid) and a red (dashed) line. The trajectory is guaranteed to satisfy specification ϕ_1.

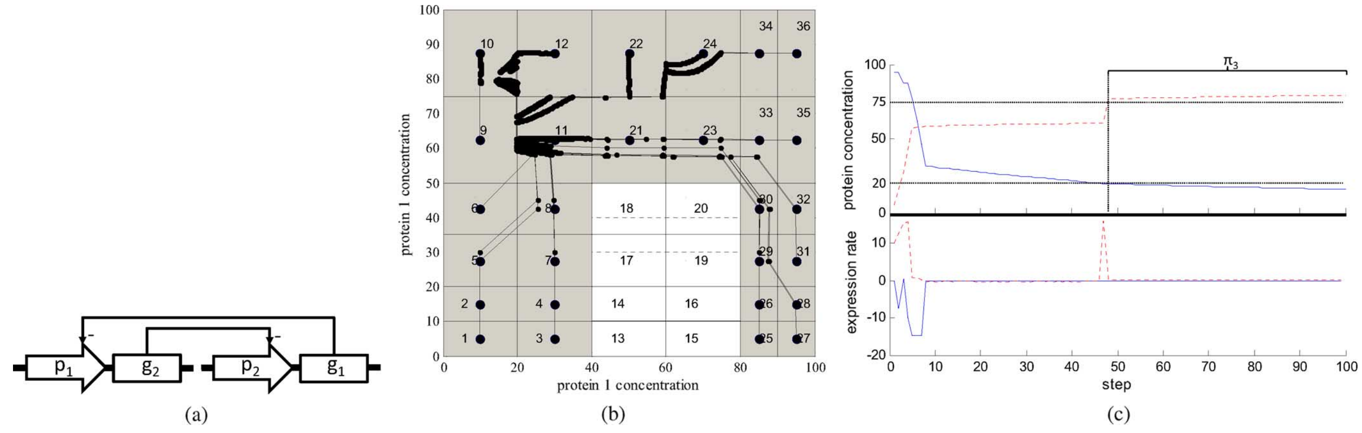

Fig. 4. (a) Schematic representation of the genetic toggle switch from [43]. The system consist of two promoters and two genes, coding for proteins that mutually repress each other, and acts as a switch allowing only one of the genes to be expressed depending on initial conditions. (b) Control strategies guaranteeing the satisfaction of specification $\phi_2 = \Diamond\Box\pi_3 \wedge \Box\neg\pi_4$ are found from all shaded regions. Simulated trajectories from different regions satisfy the specification. (c) A simulated trajectory of the closed loop system is guaranteed to satisfy specification ϕ_2. Concentrations of proteins 1 and 2 and the external control over their respective expression rates are shown as a blue (solid) and a red (dashed) line.

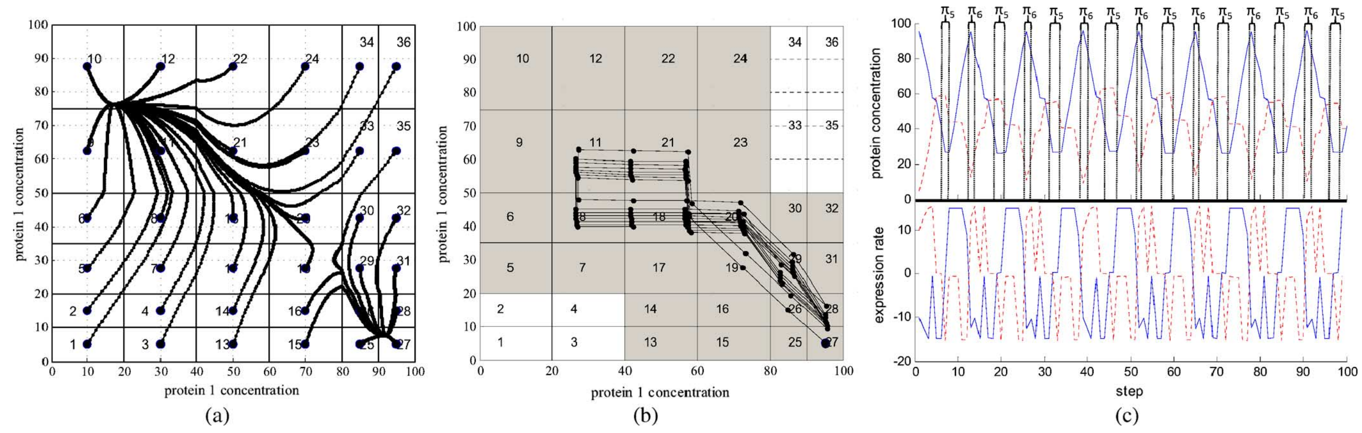

Fig. 5. (a) Trajectories of the uncontrolled PWA system go towards one of two possible stable equilibria located in regions $\mathcal{X}_{10}$ and $\mathcal{X}_{27}$ (initial states are shown as small circles). (b) Control strategies guaranteeing the satisfaction of specification $\phi_3 = \Box(\Diamond\pi_5 \wedge \Diamond\pi_6 \wedge \neg(\pi_7 \vee \pi_8))$ are found from all shaded regions. A sample satisfying simulated trajectory is shown. (c) A simulated trajectory of the closed-loop toggle switch system is guaranteed to satisfy specification ϕ_3. Protein concentrations and external expression rate control are labeled as in Fig. 4(c).

At low repressor concentration expression from the regulated promoter is maximal, expression is basal at high repressor concentration and the response is graded in between. The overall PWA model is constructed using two ramp functions (one for each promoter) which leads to nine different modes (the exact dynamics for each mode are omitted due to space constraints but are available with *conPAS2*). The PWA model captures the characteristic bi-stability of the system, where trajectories go towards one of two possible stable equilibria located in regions $\mathcal{X}_{10}$ and $\mathcal{X}_{27}$ [see Fig. 5(a)].

First, we generate a control strategy that drives the system to low concentration of protein 1 and high concentration of protein 2 while avoiding intermediate concentrations of both proteins. We define propositions $\pi_3 :=$ "protein 1 is below 20 and protein 2 is above 75" and $\pi_4 :=$ "protein 1 is above 40 and below 80 and protein 2 is above 20 and below 50" which can be expressed as disjunctions of regions from L as $\pi_3 := 10$ and $\pi_4 := 17 \vee \ldots \vee 20$. We are interested in specification

$$\phi_2 = \Diamond\Box\pi_3 \wedge \Box\neg\pi_4 \tag{16}$$

which cannot be translated into a deterministic Büchi automaton. Satisfying control strategies where found from all regions expect 13 ,. . ., 20 when the required robustness was set to $\epsilon = 5e^{-2}$ [Fig. 4(b)]. If stuttering inputs are not characterized as described in Section VII, satisfying control strategies are identified for region 10 only. A simulated trajectory of the closed loop system is shown in Fig. 4(c) and computation times are given as case study #2 in Table I.

Next, we generate a control strategy forcing the system to oscillate between states where the concentration of one protein is high and the other is low and vice versa, while states where the concentrations of both proteins are high or low are never reached. We define propositions $\pi_5 :=$ "protein 1 is below 40 and protein 2 is above 50," $\pi_6 :=$ "protein 1 is above 80 and protein 2 is below 20," $\pi_7 :=$ "protein 1 is below 40 and protein 2 is below 20," and $\pi_8 :=$ "protein 1 is above 80 and protein 2 is above 50," which can be expressed as disjunctions of regions from L as $\pi_5 := 9 \vee \ldots \vee 12$, $\pi_6 := 25 \vee \ldots \vee 28$, $\pi_7 := 1 \vee \ldots \vee 4$, and $\pi_8 := 33 \vee \ldots \vee 36$. The specification we consider is

$$\phi_3 = \Box\left(\Diamond\pi_5 \wedge \Diamond\pi_6 \wedge \neg(\pi_7 \vee \pi_8)\right). \tag{17}$$

Satisfying control strategies where found from all regions expect 1,. . ., 4 and 33,. . .,36 when the required robustness was set to $\epsilon = 5e^{-2}$ [Fig. 5(b)]. If stuttering inputs are not characterized as described in Section VII, no satisfying control strategies are identified. A simulated trajectory of the closed loop system is shown in Fig. 5(c) and computation times are given as case study #3 in Table I.

X. Conclusion

We described a computational framework for automatic generation of feedback control strategies for discrete-time, continuous-space PWA systems from rich specifications given as LTL formulas over polytopic regions in the state space. Our approach consists of two main steps: 1) abstracting the original infinite PWA control system to a finite control system, and 2) generating a control strategy for the finite control system from the LTL specification. For the latter, we use ideas from temporal logic games and automata theoretic model checking to develop an algorithm that provides a complete solution and incorporates information about stuttering behavior that arises during the abstraction process but can be characterized. The particular approach to the construction of the abstraction guarantees that a control strategy generated for the finite control system can be easily transformed into a control strategy for the initial PWA. While provably correct, the overall solution is conservative and computationally expensive.

Appendix

A. Proof of Proposition 1

Note that the set defined in (5) can be equivalently written as

$$\Sigma^l = \{u \in \mathcal{U} | \forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}\} \tag{18}$$

Let $u \in \mathcal{U}$ such that $\forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}$. Then,

$$\begin{aligned} &\forall x \in \mathcal{X}_l, H(A_l x + B_l u + c_l) < K \Rightarrow \\ &\Rightarrow \forall x \in \mathcal{X}_l, HB_l u < K - H(A_l x + c_l) \Rightarrow \\ &\Rightarrow \forall v \in \mathcal{V}(X_l), HB_l u < K - H(A_l v + c_l). \end{aligned}$$

Let $u \in \mathcal{U}$ such that $\forall v \in \mathcal{V}(\mathcal{X}_l), HB_l u < K - H(A_l v + c_l)$. Then, $\forall v \in \mathcal{V}(X_l), A_l v + B_l u + c_l \in \mathcal{X}$. Let $m = |\mathcal{V}(\mathcal{X}_l)|, x = \Sigma_{i=1}^m \lambda_i v_i$, where $v_i \in \mathcal{V}(\mathcal{X}_l), 0 < \lambda_i < 1$ for all $i = 1, \ldots, m$ and $\Sigma_{i=1}^m \lambda_i = 1$. Then,

$$\begin{aligned} A_l x + B_l u + c_l &= A_l \Sigma_{i=1}^m \lambda_i v_i + B_l u + c_l \\ &= \Sigma_{i=1}^m \lambda_i (A_l v_i + B_l u + c_l) \in \mathcal{X} \Rightarrow \\ &\Rightarrow \forall x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}. \end{aligned}$$

B. Proof of Proposition 2

The set defined in (10) is a polytope with the V-representation given in (11) Let $\exists x \in \mathcal{X}_l$ such that $A_l x + \hat{x} + c_l \in \mathcal{X}'$. Let $m = |\mathcal{V}(\mathcal{X}_l)|$ and $x = \Sigma_{i=1}^m \lambda_i v_i$, where $0 < \lambda_i < 1$ for all $i = 1, \ldots, m$ and $\Sigma_{i=1}^m \lambda_i = 1$. Let $n = |\mathcal{V}(\mathcal{X}')|$ and $x' = \Sigma_{j=1}^n \mu_j v'_j$, where $0 < \mu_j < 1$ for all $j = 1, \ldots, n$ and $\Sigma_{j=1}^n \mu_j = 1$. Then,

$$\begin{aligned} A_l \Sigma_{i=1}^m \lambda_i v_i + \hat{x} + c_l &= \Sigma_{j=1}^n \mu_j v'_j \Rightarrow \\ \Rightarrow \hat{x} &= \Sigma_{j=1}^n \mu_j v'_j - A_l \Sigma_{i=1}^m \lambda_i v_i - c_l \\ &= \Sigma_{i=1}^m \Sigma_{j=1}^n \lambda_i \mu_j \left(v'_j - (A_l v_i + c_l)\right) \Rightarrow \\ \Rightarrow \hat{x} &\in \text{hull}\{v' - (A_l v + c_l) | v \in \mathcal{V}(\mathcal{X}_l), \\ &\qquad v' \in \mathcal{V}(\mathcal{X}')\}. \end{aligned}$$

Let $\hat{x} = \Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij} (v'_j - (A_l v_i + c_l))$, where $0 < \nu_{ij} < 1, i = 1, \ldots, m, j = 1, \ldots, n$ and $\Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij} = 1$. Let $\lambda_i = \Sigma_{j=1}^m \nu_{ij}$ and $\mu_j = \Sigma_{i=1}^n \nu_{ij}$. Of course, $0 < \lambda_i < 1$ for all $i = 1, \ldots, m$, $0 < \mu_j < 1$ for all $j = 1, \ldots, n$ and $\Sigma_{i=1}^m \lambda_i = \Sigma_{j=1}^n \mu_j = \Sigma_{i=1}^m \Sigma_{j=1}^n \nu_{ij} = 1$. Then, for $x = \Sigma_{i=1}^m \lambda_i v_i$ and $x' = \Sigma_{j=1}^n \mu_j v'_j$ we have $A_l x + \hat{x} + c_l = x'$ and therefore $\exists x \in \mathcal{X}_l$ such that $A_l x + \hat{x} + c_l \in \mathcal{X}'$.

To conclude the proof of Proposition 2, let H, K be the matrices in the H-representation of the set defined in (10) and note that the set defined in (9) can be written as

$$U^{\mathcal{X}_l \to \mathcal{X}'} = \{u \in \mathcal{U} | \exists x \in \mathcal{X}_l, A_l x + B_l u + c_l \in \mathcal{X}'\}. \tag{19}$$

C. Proof of Proposition 3

From (3) and (4) we have

$$\begin{aligned} U_l^{L'} &= \{u \in \Sigma_e | \forall l' \in L', Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l'} \neq \emptyset, \\ &\quad \forall l'' \notin L', Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l''} = \emptyset\} \\ &= \{u \in \Sigma_e | \forall l' \in L', Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l'} \neq \emptyset\} \setminus \\ &\quad \times \{u \in \Sigma_e | \exists l'' \notin L', Post_{\mathcal{T}_e}(\mathcal{X}_l, u) \cap \mathcal{X}_{l''} \neq \emptyset\} \\ &= \bigcap_{l' \in L'} U^{\mathcal{X}_l \to \mathcal{X}_{l'}} \setminus \bigcup_{l'' \notin L'} U^{\mathcal{X}_l \to \mathcal{X}_{l''}}. \end{aligned}$$

D. Proof of Proposition 4

($\Rightarrow$) Let $0 \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l | \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$. Then, there exists $x \in \mathcal{X}_l, u \in U_l^{L'}$ such that $A_l x + B_l u + c_l = x$ and a trajectory of the system produced by applying input sequence $uuu\ldots$ and starting at x remains forever inside $\mathcal{X}$. Therefore, from Def. 8, $U_l^{L'}$ is not stuttering.

($\Leftarrow$) Let $0 \notin \text{hull}\{(A_l - I)v_x + B_l v_u + c_l | \forall v_x \in \mathcal{V}(\mathcal{X}_l), \forall v_u \in \mathcal{V}(U_l^{L'})\}$. From the separating hyperplane theorem it follows that there exists $a \in \mathbb{R}^N$ such that, for all $z \in \text{hull}\{(A_l - I)v_x + B_l v_u + c_l | \forall v_x \in \mathcal{V}(\mathcal{X}_l)), a^{\mathcal{T}} z > 0$. Then, any trajectory of the system originating in $\mathcal{X}_l$ and produced by input word $u_1 u_2 u_3 \ldots$, where $u_i \in U_l^{L'}$ will have a positive displacement along the direction of $a^{\mathcal{T}}$ at every step. Since $\mathcal{X}_l$ is bounded, all trajectories will leave it in a finite number of steps and, therefore, $U_l^{L'}$ is stuttering.

E. Proof of Proposition 5

$$\begin{aligned} \hat{U}_l^{L'} &= \Big\{u \in U_l^{L'} | \forall v_x \in \mathcal{V}(\mathcal{X}_l), \\ &\qquad a^T B_l u > -a^T\left((A_l - I)v_x - c_l\right)\Big\} \\ &= \Big\{u \in U_l^{L'} | \forall v_x \in \mathcal{V}(\mathcal{X}_l), \\ &\qquad a^T\left((A_l - I)v_x + B_l u + c_l\right) > 0\Big\} \\ &\Rightarrow 0 \notin \text{hull}\Big\{(A_l - I)v_x + B_l v_u + c_l | \forall v_x \in \mathcal{V}(\mathcal{X}_l), \\ &\qquad \forall v_u \in \mathcal{V}\left(\hat{U}_l^{L'}\right)\Big\} \end{aligned}$$

which, from Prop. 4, guarantees that $\hat{U}_l^{L'}$ is stuttering.

References

[1] E. M. Clarke, D. Peled, and O. Grumberg, *Model Checking*. Cambridge, MA: MIT Press, 1999.

[2] J. M. Davoren, V. Coulthard, N. Markey, and T. Moor, "Non-deterministic temporal logics for general flow systems," in *Proc. 7th Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, 2004, pp. 280–295, ser. LNCS.

[3] P. Tabuada and G. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Trans. Autom. Control*, vol. 51, no. 12, pp. 1862–1877, Dec. 2006.

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

[5] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proc. 44th IEEE Conf. Decision Control (CDC)*, 2005, pp. 4885–4890.

[6] H. Kress-Gazit, D. Conner, H. Choset, A. Rizzi, and G. Pappas, "Courteous cars," *IEEE Robot. Autom. Mag.*, vol. 15, no. 1, pp. 30–38, Mar. 2008.

[7] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robot. Autom. Mag., Special Iss. Grand Challenges for Robotics*, vol. 14, no. 1, pp. 61–71, Mar. 2007.

[8] M. Antoniotti, F. Park, A. Policriti, N. Ugel, and B. Mishra, "Foundations of a query and simulation system for the modeling of biochemical and biological processes," in *Proc. Pacific Symp. Biocomput. (PSB)*, 2003, pp. 116–127.

[9] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page, and D. Schneider, "Validation of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in Escherichia coli," *Bioinformatics*, vol. 21, no. Suppl. 1, pp. i19–i28, 2005.

[10] G. Batt, C. Belta, and R. Weiss, "Temporal logic analysis of gene networks under parameter uncertainty," *IEEE Trans. Circuits Syst. IEEE Trans. Autom. Control, Joint Special Iss. Syst. Biol.*, vol. 53, no. 1, pp. 215–229, Jan. 2008.

[11] W. P. M. H. Heemels, B. D. Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, 2001.

[12] A. L. Juloski, W. P. M. H. Heemels, G. Ferrari-Trecate, R. Vidal, S. Paoletti, and J. H. G. Niessen, "Comparison of four procedures for the identification of hybrid systems," in *Proc. 8th Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, 2005, pp. 354–369, ser. LNCS.

[13] W. Thomas, "Infinite games and verification," in *Proc. 14th Int. Conf. Comput. Aided Verif. (CAV)*, 2002, pp. 58–64.

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

[15] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, 2003.

[16] M. Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Proc. 11th Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, 2008, pp. 287–300, ser. LNCS. Springer Berlin/Heidelberg.

[17] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, C. Tomlin and M. Greenstreet, Eds. Berlin/Heidelberg, Germany: Springer, 2002, vol. 2289, pp. 425–438.

[18] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proc. 16th ACM SIGPLAN-SIGACT Symp. Principles Program. Lang.*, New York, 1989, pp. 179–190, ser. POPL '89, ACM.

[19] S. Jiang and R. Kumar, "Supervisory control of discrete event systems with CTL* temporal logic specifications," *SIAM J. Control Optimiz.*, vol. 44, no. 6, pp. 2079–2103, 2006.

[20] S. Basu and R. Kumar, "Quotient-based control synthesis for non-deterministic plants with Mu-calculus specifications," in *Proc. 45th IEEE Conf. Decision Control*, 2006, pp. 6041–6046.

[21] F. Horn, "Streett games on finite graphs," in *Proc. 2nd Workshop Games in Design Verification (GDV)*, 2005.

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

[23] O. Păun and M. Chechik, "On closure under stuttering," *Formal Aspects Comput.*, vol. 14, no. 4, pp. 342–368, Apr. 2003.

[24] D. A. Peled and T. Wilke, "Stutter-invariant temporal properties are expressible without the next-time operator," *Inf. Process. Lett.*, vol. 63, no. 5, pp. 243–246, Sep. 1997.

[25] M. Kvasnica, P. Grieder, and M. Baotić, Multi-Parametric Toolbox (MPT), 2004. [Online]. Available: http://control.ee.ethz.ch/mpt/

[26] A. Bemporad, Hybrid Toolbox—User's Guide, 2004. [Online]. Available: http://www.ing.unitn.it/bemporad/hybrid/toolbox

[27] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. 47th IEEE Conf. Decision Control (CDC)*, 2008, pp. 2117–2122.

[28] M. Mazo, A. Davitian, and P. Tabuada, "PESSOA: A tool for embedded controller synthesis," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, T. Touili, B. Cook, and P. Jackson, Eds. Berlin/Heidelberg, Germany: Springer, 2010, vol. 6174, pp. 566–569.

[29] G. Frehse, S. K. Jha, and B. H. Krogh, "A counterexample-guided approach to parameter synthesis for linear hybrid automata," in *Proc. 11th Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, M. Egerstedt and B. Mishra, Eds., 2008, pp. 187–200, ser. LNCS.

[30] B. Yordanov and C. Belta, "Formal analysis of discrete-time piecewise affine systems," *IEEE Trans. Autom. Control*, vol. 55, no. 12, pp. 2834–2841, Dec. 2010.

[31] B. Yordanov and C. Belta, "Parameter synthesis for piecewise affine systems from temporal logic specifications," in *Proc. 11th Int. Conf. Hybrid Syst.: Comput. Control (HSCC)*, M. Egerstedt and B. Mishra, Eds., 2008, pp. 542–555, ser. LNCS, Springer Berlin/Heidelberg.

[32] B. Yordanov and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," in *Proc. 48th IEEE Conf. Decision Control*, Shanghai, China, Dec. 2009, pp. 3182–3187.

[33] J. Tůmová, B. Yordanov, C. Belta, I. Černá, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *Proc. 49th IEEE Conf. Decision Control*, Atlanta, GA, Dec. 2010, pp. 4230–4235.

[34] K. Fukuda, cdd/cdd+ Package, 1997. [Online]. Available: http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html

[35] S. Safra, "On the complexity of omega-automata," in *Proc. 29th Annu. IEEE Symp. Foundat. Comput. Sci. (FOCS)*, 1988, pp. 319–327.

[36] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theoret. Comput. Sci.*, vol. 363, no. 2, pp. 182–195, 2006.

[37] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Proc. IEEE Symp. Logic in Comput. Sci. (LICS)*, Washington, DC, 1986, pp. 332–344, IEEE Computer Society.

[38] E. A. Emerson, "Automata, tableaux and temporal logics (extended abstract)," in *Proc. Conf. Logic of Programs*, 1985, pp. 79–88.

[39] J. C. Mitchell, "Discrete uniform sampling of rotation groups using orthogonal images," *SIAM J. Sci. Comput.*, vol. 30, no. 1, pp. 525–547, 2007.

[40] N. Piterman and A. Pnueli, "Faster solutions of Rabin and Streett games," in *Proc. Logic in Comput. Sci., Symp.*, 2006, vol. 0, pp. 275–284.

[41] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, E. Emerson and K. Namjoshi, Eds. Berlin/Heidelberg, Germany: Springer, 2006, vol. 3855, pp. 364–380.

[42] J. Thomas, S. Olaru, J. Buisson, and D. Dumu, "Robust model predictive control for piecewise affine systems subject to bounded disturbances," in *Proc. 2nd IFAC Conf. Anal. Design of Hybrid Syst.*, Alghero, Italy, 2006.

[43] T. Gardner, C. Cantor, and J. Collins, "Construction of a genetic toggle switch in *Escherichia coli*," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

Boyan Yordanov (M'11) received the B.A. degrees in biochemistry and computer science from Clark University, Worcester MA, in 2005 and the M.S. and Ph.D. degrees in biomedical engineering from Boston University, Boston MA, in 2009 and 2011, respectively.

He was a Postdoctoral Researcher in the Department of Mechanical Engineering, Boston University, in 2011 and is currently a Postdoctoral Researcher at the Biological Computation Group as part of the Computational Science Laboratory at Microsoft Research, Cambridge, U.K. His research interests are in the areas of analysis and design of biological systems, synthetic biology, temporal logic, formal methods, and hybrid systems.

Jana Tůmová received the B.S. and M.Sc. degrees in applied computer science and computer science from Masaryk University, Brno, Czech Republic, in 2006 and 2009, respectively. She is currently pursuing the Ph.D. degree in computer science at Masaryk University. Her research interests include formal methods, temporal logics, model checking, and controller synthesis.

Ivana Černá received the M.Sc. and Ph.D. degrees in computer science from Comenius University, Bratislava, Slovak Republic, in 1986 and 1992, respectively.

She is a Professor at Faculty of Informatics, Masaryk University, Brno, Czech Republic. Her research interests include theory of communicating and parallel systems, formal verification and verification tools, algorithm design, and analysis. She is a coauthor of algorithms implemented in a parallel and distributed verification tool DiVinE.

Jiří Barnat received the M.Sc. and Ph.D. degrees in computer science from Masaryk University, Brno, Czech Republic, in 2000 and 2005, respectively.

He is an Associate Professor at Faculty of Informatics, Masaryk University. His research interests include parallel algorithms, parallel and distributed methods in formal verification, and platform dependent algorithm engineering. He is currently leading the ParaDiSe research group at the Faculty of Informatics. He is a coauthor of parallel and distributed verification tool DiVinE.

Calin Belta (M'03–SM'11) received the B.S. and M.Sc. degrees in control and computer science from the Technical University of Iasi, Iasi, Romania, the M.Sc. degree in electrical engineering from Louisiana State University, Baton Rouge, and the M.Sc. and Ph.D. degrees in mechanical engineering from the University of Pennsylvania, Philadelphia.

He is currently an Associate Professor of Mechanical Engineering, Systems Engineering, and Bioinformatics at Boston University, Boston, MA. His research interests include analysis and control of hybrid systems, motion planning and control, and bio-molecular networks.

Dr. Belta is an Associate Editor for the *SIAM Journal on Control and Optimization* (SICON). He received the AFOSR Young Investigator Award in 2008 and the NSF CAREER Award in 2005.