49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

A Symbolic Approach to Controlling Piecewise Affine Systems

Jana Tůmová, Boyan Yordanov, Calin Belta, Ivana Černá, and Jiří Barnat

***Abstract*— We present a computational framework for automatic synthesis of a feedback control strategy for a piecewise affine (PWA) system from a specification given as a Linear Temporal Logic (LTL) formula over an arbitrary set of linear predicates in its state variables. First, by defining 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, we develop an algorithm to generate a control strategy for the finite abstraction. While provably correct and robust to small perturbations in both state measurements and applied inputs, the overall procedure is conservative and expensive. The proposed algorithms have been implemented and are available for download. Illustrative examples are included.**

I. Introduction

Temporal logics and model checking [5] are customarily used for specifying and verifying the correctness of digital circuits and computer programs. However, 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 [6], control of linear systems from temporal logic specifications [21], [15], task specification and controller synthesis in mobile robotics [16], [4] and specification and analysis of qualitative behavior of genetic networks [2], [3].

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 other classes of hybrid systems [9]. In addition, there exist techniques for the identification of such models from experimental data (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 its state variables, find a feedback control strategy such that all trajectories of the closed loop system 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 Rabin games [22] and LTL model checking [5], we develop an algorithm to generate a control strategy for the finite abstraction.

This work was partially supported by grants ARO W911NF-09-1-0088, NSF CNS-0834260, and AFOSR FA9550-09-1-020 at Boston University and by grants GA201/09/1389, and GA201/09/P497 at Masaryk University. J. Tůmová (xtumova@fi.muni.cz) is with Masaryk University and Boston University. B. Yordanov and C. Belta ({yordanov,cbelta}@bu.edu) are with Boston University. I. Černá and J. Barnat ({cerna,barnat}@fi.muni.cz) are with Masaryk University.

The contribution of this work is three-fold. First, it provides a general and fully automatic framework for controlling finite nondeterministic transition systems from specifications given as arbitrary LTL formulas. This extends our results from [18], where completeness was only guaranteed for specifications given in a fragment of LTL generated by deterministic Büchi automata. This significantly increases the expressivity of the specification language. Second, by dealing with the stuttering phenomenon inherent in the finite abstraction and maintaining both time specific and time abstract information about the system, it reduces the conservativeness of the approach that we recently proposed for the above problem in [25], while expressivity is not sacrificed. Third, it seamlessly combines the abstraction and control procedures into a computational framework allowing for fully automatic generation of PWA feedback control strategies from high-level, rich LTL specifications. The framework was implemented as a freely downloadable tool `conPAS2`.

This paper can be seen in the context of literature focused on the construction of finite quotients of infinite systems (see [1] for an earlier review), and is related to [19], [21], [15]. The embedding into transition systems is inspired from [19], [21], where the existence of bisimulation quotients and control strategies under the assumption of controllability for linear systems is characterized. In this work, we focus instead on developing algorithmic procedures for the computation of quotients and control strategies for the more general class of PWA systems. This paper is also related to literature focused on controlling finite systems, such as discrete-event systems, from temporal logic specifications such as CTL* formulas [11]. Although similar, our approach to controlling nondeterministic transition systems can handle information about the stuttering behavior that arises during the construction of finite quotients. While controllers for PWA systems can be synthesized using other tools [17], our approach allows for more general, temporal logic specifications. The related problem of temporal logic control of Mixed Logical Dynamical (MLD) systems has been considered in [13] by representing LTL specifications as mixed-integer linear constraints but a finite horizon assumption is imposed. This paper extends recent results on formal analysis of PWA systems [8], [26] to a control framework.

Due to space limitations, some proofs and additional details are omitted but made available as a technical report [23].

II. Notation and Preliminaries

Given a set Q, we use $|Q|$, 2^Q, Q^+, and Q^ω to denote its cardinality, powerset, and sets of nonempty finite and infinite sequences of elements from Q, respectively.

978-1-4244-7746-3/10/$26.00 ©2010 IEEE

Definition 1: A nondeterministic transition system is a tuple $\mathcal{T} = (Q, \Sigma, \delta, O, o)$, where Q and Σ are sets of states and inputs, $\delta : Q \times \Sigma \to 2^Q$ is a 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 $\Sigma^q = \{\sigma \in \Sigma \mid \delta(q, \sigma) \neq \emptyset\}$. A transition $\delta(q, \sigma)$ is *deterministic* if $|\delta(q, \sigma)| = 1$ and transition system $\mathcal{T}$ is deterministic if for all $q \in Q$ and all $\sigma \in \Sigma^q$, $\delta(q, \sigma)$ is deterministic. $\mathcal{T}$ is *finite* if both Q and Σ are finite. $\mathcal{T}$ is *non-blocking* if, for every state $q \in Q$, $\Sigma^q \neq \emptyset$. In this work, we consider only non-blocking transition systems.

An *input word* of the system is defined as an infinite sequence $\sigma_0\sigma_1 \ldots \in \Sigma^\omega$. A *trajectory* of $\mathcal{T}$ produced by input word $\sigma_0\sigma_1 \ldots$ and originating at state $q_0 \in Q$ is an infinite sequence $q_0q_1 \ldots$ with the property that $q_{k+1} \in \delta(q_k, \sigma_k)$, for all $k \geq 0$. We denote the set of all trajectories of $\mathcal{T}$ originating at q by $\mathcal{T}(q)$ (similarly, $\mathcal{T}(Q') = \cup_{q' \in Q'} \mathcal{T}(q')$ denotes the set of all trajectories of $\mathcal{T}$ originating in $Q' \subseteq Q$). A trajectory $q_0q_1 \ldots$ defines a *word* $o(q_0)o(q_1) \ldots$.

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) and the sets of all trajectories and words of the closed loop $\mathcal{T}$ by $\mathcal{T}(Q_0, \Omega)$ and $\mathcal{L}_{\mathcal{T}}(Q_0, \Omega)$, respectively. Any trajectory $q_0q_1 \ldots \in \mathcal{T}(Q_0, \Omega)$ satisfies $q_0 \in Q_0$ and $q_{k+1} \in \delta(q_k, \sigma_k)$, where $\sigma_k = \Omega(q_0, \ldots, q_k)$, for all $k \geq 1$.

To specify temporal properties of trajectories of transition systems (and PWA systems, as it will become clear later) we use Linear Temporal Logic [5]. Informally, LTL formulas are inductively defined over a set of observations O, by using the standard Boolean operators and temporal operators $\bigcirc$ ("next"), $\cup$ ("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. We denote by $\mathcal{L}_\phi$ the language of words that satisfy the formula ϕ.

A Rabin automaton is a tuple $\mathcal{R} = (S, S_0, O, \delta_{\mathcal{R}}, F)$, where S is a set of states, $S_0 \subseteq S$ is the set of initial states, O is the input alphabet, $\delta_{\mathcal{R}} : S \times O \to 2^S$ 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 $w = o_0o_1 \ldots \in O^\omega$ is a sequence $\rho = s_0s_1 \ldots$, 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. A run ρ is accepting if $inf(\rho) \cap G_i \neq \emptyset \wedge inf(\rho) \cap B_i = \emptyset$ for some $i \in \{1, \ldots, n\}$. An input word w is accepted by an automaton if some run over w is accepting. We denote by $\mathcal{L}_{\mathcal{R}}$ the language of words accepted by $\mathcal{R}$.

Given an LTL formula ϕ, one can build a deterministic Rabin automaton $\mathcal{R}$ with $2^{2^{O(|\phi| \cdot \log |\phi|)}}$ states and $2^{O(|\phi|)}$ pairs in its acceptance condition, such that $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_\phi$ [20]. The translation can be done using standard tools, *e.g.*, `ltl2dstar` [14].

III. PROBLEM FORMULATION AND APPROACH

Let $\mathcal{X}, \mathcal{X}_l$, $l \in L$ be a set of 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:

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

where, at each time step $k = 0, 1, \ldots$, $x_k \in \mathbb{R}^N$ 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 $l \in L$.

We assume that 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. Intuitively, a trajectory of the system produces a word by listing the index of the polytope visited at each step (*e.g.*, trajectory $x_0x_1x_2 \ldots$ satisfying $x_0, x_1 \in \mathcal{X}_{l_1}$ and $x_2 \in \mathcal{X}_{l_2}$ for some $l_1, l_2 \in L$ will produce word $l_1l_1l_2 \ldots$). We assume that polytope $\mathcal{X}$ is an invariant for all trajectories of the system (in [25] we showed that this can always be guaranteed through polyhedral control constraints) and, thus, only infinite words are produced. Then, such words can be checked against the satisfaction of an LTL formula over L.

We consider the following problem:

Problem 1: Given a PWA system (1) and an LTL formula ϕ over L, find a control strategy, such that all trajectories of the closed loop system satisfy ϕ, while remaining within $\mathcal{X}$.

In order to complete the formulation of Problem 1, we need to formalize the definitions of a control strategy for a PWA system (1) and satisfaction of LTL formulas by trajectories of (1). We do this through an embedding into a transition system, for which both LTL satisfaction and a control strategy are clearly defined.

Definition 3: The embedding transition system $\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$ for system (1) is: $Q_e = \bigcup_{l \in L} \mathcal{X}_l$, $\Sigma_e = \mathcal{U}$, $\delta_e(x, u) = \{x'\}$ 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 4: Trajectories of system (1) originating in $Q_0 \subseteq Q_e$ satisfy formula ϕ if and only if $\mathcal{T}_e(Q_0)$ satisfies ϕ.

Problem 1 can be considered an LTL control problem, where we seek a control strategy (Q_0, Ω) (Def. 2) for the infinite, deterministic transition system $\mathcal{T}_e$. A preliminary solution to Problem 1 was presented in [25], where we constructed control transition system $\mathcal{T}_c$ as a finite abstraction for $\mathcal{T}_e$ and showed that a control strategy generated for $\mathcal{T}_c$ can be adapted for $\mathcal{T}_e$ (we summarize those results in Sec. IV). We treated the cases when $\mathcal{T}_c$ was deterministic and non-deterministic separately and allowed full LTL expressivity for a deterministic $\mathcal{T}_c$ (which corresponds to a conservative approach to the abstraction process). For a nondeterministic $\mathcal{T}_c$, the expressivity was restricted to a fragment of LTL generated by a deterministic Büchi automaton. This solution was conservative, since not all LTL formulas can be translated into deterministic Büchi automata (*e.g.*, $\Diamond\Box\phi$ for any LTL formula ϕ). The stuttering phenomenon (self transitions at a state of $\mathcal{T}_c$ that can be taken infinitely in $\mathcal{T}_c$ but do not correspond to real trajectories of $\mathcal{T}_e$), which is also related to the well known Zeno behavior, was an additional source of conservativeness in [25].

In this paper, by developing a complete control strategy for nondeterministic transition systems from full LTL specifications in Sec. V and by characterizing and dealing with stuttering phenomena in Sec. VI, we significantly (1) reduce the conservativeness of our previous approach and (2) increase the expressivity of the specification language.

Remark 1: We make some simplifying assumptions in the formulation of Problem 1 that might seem restrictive. First, we capture only the reachability of open full dimensional polytopes in the semantics of the embedding. 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 originating and remaining in such sets are therefore of no interest. Trajectories originating 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 priory*. However, arbitrary linear inequalities can be accommodated by including additional polytopes, in which case the system will have the same dynamics in several modes.

IV. Control Transition System

In this section, we summarize the construction of control transition system $\mathcal{T}_c = (Q_c, \Sigma_c, \delta_c, O_c, o_c)$ for the infinite $\mathcal{T}_e = (Q_e, \Sigma_e, \delta_e, O_e, o_e)$ that was presented in [25]. Then, we show how a control strategy for $\mathcal{T}_c$ can be adapted to $\mathcal{T}_e$.

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 if and only if $o_e(x) = o_e(x')$. The equivalence relation naturally 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 Σ_e 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 $u \in \Sigma_e, x \in \mathcal{X}_l$ and $x' \in \mathcal{X}'$ such that $x' = \delta_e(x,u)$. 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. Note that $\mathcal{T}_e/_\sim$ is, in general nondeterministic, even though $\mathcal{T}_e$ is deterministic.

For each state $l \in Q_e/_\sim$, we define an equivalence relation $\approx_l$ over the set of inputs Σ_e as $(u_1,u_2) \in \approx_l$ if and only if $\delta_{e\sim}(l,u_1) = \delta_{e\sim}(l,u_2)$ (*i.e.*, inputs u_1 and u_2 are equivalent at l if they produce the same transitions in $\mathcal{T}_e/_\sim$). Let $U_l^{L'} = \{u \in \Sigma_e \mid l \in L, L' \in 2^{Q_e/_\sim}, \delta_{e\sim}(l,u) = L'\}$ denote the equivalence classes of Σ_e in the partition induced by $\approx_l$. In [25] we showed that the equivalence classes $U_l^{L'}$ can be computed using polyhedral operations and can be represented as finite unions of polytopes. Let $u_l^{L'} \in U_l^{L'}$ be an input such that $\forall u \in \Sigma_e$ it holds that $d(u_l^{L'},u) < \varepsilon \Rightarrow u \in U_l^{L'}$, where $d(u_1,u_2)$ denotes the Euclidean distance in $\mathbb{R}^M$ and ε is a predefined parameter specifying the robustness of the control strategy. In other words, $u_l^{L'} \in U_l^{L'}$ is the center of a sphere with a radius larger than ε, inscribed in $U_l^{L'}$ and input $u_l^{L'}$ is available at a state l in $\mathcal{T}_c$ (*i.e.,* $u_l^{L'} \in \Sigma_c^l$) if such a sphere can be computed[1], which clearly induces transition $\delta_c(l,u_l^{L'}) = L'$. In general, it is possible that at a given state l, $\Sigma_c^l = \emptyset$, (*i.e.*, state l is blocking). Such states are recursively removed from the system together with their incoming transitions and therefore $Q_c \subseteq L$. Following from the construction outlined above, $\mathcal{T}_c$ has a finite set of states and inputs.

Definition 5: 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 $q_0 \dots q_k$ where $q_0 \in Q_0$, the control function is defined as $\Omega(q_0 \dots q_k) = \Omega^c(o(q_0) \dots o(q_k))$.

Proposition 1: Given a control strategy (Q_0^c, Ω^c) for $\mathcal{T}_c$ translated as a control strategy (Q_0, Ω) for $\mathcal{T}_e$, $\mathcal{L}_{\mathcal{T}_e}(Q_0,\Omega) \subseteq \mathcal{L}_{\mathcal{T}_c}(Q_0^c,\Omega^c)$, which implies that if $\mathcal{T}_c(Q_0^c,\Omega^c)$ satisfies an arbitrary LTL formula ϕ, then so does $\mathcal{T}_e(Q_0,\Omega)$.

Following from Prop. 1, which we proved in [23], a control strategy for $\mathcal{T}_c$ can be adapted to the infinite $\mathcal{T}_e$. The control strategy is robust with respect to perturbations in the measured state (*i.e.*, it depends on the observation of a state of $\mathcal{T}_e$ rather than the state itself). It is also robust to perturbations in the applied inputs, bounded by the predefined parameter ε.

V. LTL Control for Finite Transition Systems

In this section, we consider the following problem:

Problem 2: Given a finite (nondeterministic) transition system $\mathcal{T}$ from Def. 1 (such as the control transition system $\mathcal{T}_c$ from Sec. IV) and an LTL formula ϕ, find a control strategy (Def. 2), such that all trajectories of the closed loop system satisfy ϕ.

We first reformulate Problem 2 as a Rabin game and then adapt the solution to the Rabin game as a control strategy for $\mathcal{T}$. As it will become clear later, the control strategy takes the form of a "feedback automaton", which reads the current state of $\mathcal{T}$ and outputs the input to be applied at that state.

Given a finite transition system $\mathcal{T} = (Q, \Sigma, \delta, O, o)$ and an LTL formula ϕ over O we can translate ϕ into a deterministic Rabin automaton $\mathcal{R} = (S, S_0, O, \delta_\mathcal{R}, F)$ (see Sec. II) and construct the *product automaton* $\mathcal{P} = (S_\mathcal{P}, S_{\mathcal{P}0}, \Sigma, \delta_\mathcal{P}, F_\mathcal{P})$, 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 $\delta_\mathcal{P}((q,s),\sigma) = \{(q',s') \in S_\mathcal{P} \mid q' \in \delta(q,\sigma)$, and $s' = \delta_\mathcal{R}(s,o(q))\}$, and $F_\mathcal{P} = \{(Q \times G_1, Q \times B_1), \dots, (Q \times G_n, Q \times B_n)\}$ 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 $\rho_\mathcal{P} = (q_0,s_0)(q_1,s_1)\dots$ of $\mathcal{P}$ can be projected into a trajectory $q_0q_1\dots$ of $\mathcal{T}$, such that the word $o(q_0)o(q_1)\dots$

[1] in [25] we computed the inscribed spheres of all polytopes from a set $U_l^{L'}$. Then, $u_l^{L'}$ was the center of the sphere with the largest radius if it was greater than ε and otherwise we considered $U_l^{L'}$ to be empty

is accepted by $\mathcal{R}$ (*i.e.*, satisfies ϕ) and vice versa [24]. This allows us to reduce Problem 2 to finding a control strategy $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$ for $\mathcal{P}$, such that each run of the closed loop $\mathcal{P}$ satisfies the Rabin acceptance condition $F_{\mathcal{P}}$ [2]. 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 [7], the control function is simply a map $\pi_{\mathcal{P}} : S_{\mathcal{P}} \to \Sigma$.

Rabin games can be solved by standard algorithms. In this paper we follow the approach by Horn [10], which can be adapted to deal with the stuttering behavior as we will explain in Sec. VI. The basic step of the recursive algorithm is attractor construction. A protagoninst's (or adversary's) attractor of a set $S' \subseteq S_{\mathcal{P}}$ is defined as a set of states from which the protagonist (or the adversary, respectively) can enforce a visit to S'.

By solving the Rabin game we generate a control strategy $(W_{\mathcal{P}0}, \pi_{\mathcal{P}})$ for $\mathcal{P}$. In order to complete the solution to Problem 2, 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 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

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

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.

The solution to Problem 2 allows us to generate a control strategy for the finite $\mathcal{T}_c$, which can then be adapted to a control strategy for the infinite $\mathcal{T}_e$ (Def. 5). This provides a solution to Problem 1.

[2]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.

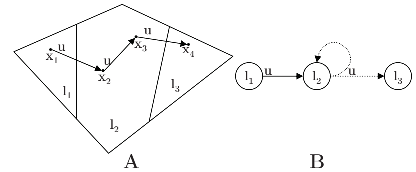


Fig. 1: A trajectory remaining forever in state l_2 exists in the finite abstraction **B**), although such behavior is not necessarily possible in the concrete system **A**)

VI. Stuttering phenomenon

In order to generate a control strategy for an infinite transition system such as $\mathcal{T}_e$ (Problem 1) we described the construction of a finite control abstraction $\mathcal{T}_c$ in Sec. IV. However, due to *spurious trajectories* (*i.e.*, trajectories of $\mathcal{T}_c$ not present in $\mathcal{T}_e$) 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. In [26] we eliminated spurious trajectories through state refinement but the states of $\mathcal{T}_c$ cannot be refined since a control strategy cannot differentiate between states having the same observation. In the following, we present an alternative approach for reducing this conservatism.

We characterize only a specific class of spurious trajectories, which we introduce through an example (Fig. 1). 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. 1-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 Sec. IV it follows that $l_2 \in \delta_c(l_1, u)$ and $\{l_2, l_3\} \subseteq \delta_c(l_2, u)$ (Fig. 1-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 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 using this additional information during the construction of the control strategy for $\mathcal{T}_c$.

Definition 6: 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. 6 we identify a stuttering subset $\Sigma_c^{ls} \subseteq \Sigma_c^l$ of the inputs available at a state $l \in Q_c$. Let $u = u_l^{L'} \in \Sigma_c^l$ for some $L' \in 2^{Q_c}$ be an input of $\mathcal{T}_c$ computed as described in Sec. IV. 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 $uuu\ldots$). We denote the rest of the inputs by $\Sigma_c^{lu} = \Sigma_c^l \setminus \Sigma_c^{ls}$.

Note that while we only characterize spurious infinite 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 using polyhedral operations as described in Prop. 2.

Proposition 2: 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 $0 \notin (A_l - I_N)\mathcal{X}_l \oplus B_l U_l^{L'} + c_l$, where I_N is the identity matrix and $\oplus$ denotes the Minkowski (set) sum.

A proof of Prop. 2 is available in [23].

The algorithm by Horn [10] from Sec. V 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 Sec. V 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 Sec. V, we need to account for the fact that the adversary cannot take transitions under the same stuttering input infinitely many times in a row. As a result, the construction of the control strategy is still performed using Horn's algorithm and only the computations of the attractors are modified as follows.

Let $l \in Q_c$ and $u \in \Sigma_c^{ls}$ be a state and a stuttering input of $\mathcal{T}_c$ (Def. 6). We are interested in *edge* (s,u,s') of transition $\delta_{\mathcal{P}}(s,u) = S'$, where $\alpha(s) = l$ and $s' \in S'$. Edge (s,u,s') is called *u-nontransient* edge if $\alpha(s) = \alpha(s') = l$ and *transient* otherwise. Note that, even though (l,u,l) is a self loop in T_c, (s,u,s') is not necessarily a self loop in $\mathcal{P}$. In addition, since there is at most one self loop at a state $l \in Q_c$ and $\mathcal{R}$ is deterministic, there is at most one u-nontransient edge leaving state s.

We refer to a sequence of edges $(s_1,u_1,s_2)(s_2,u_2,s_3)\ldots(s_{n-1},u_{n-1},s_n)$, where $s_i \neq s_j$ for any $i,j \in \{1,\ldots,n\}$ as a *simple path*, and to a simple path $(s_1,u_1,s_2)\ldots(s_{n-1},u_{n-1},s_n)$ followed by (s_n,u_n,s_1) as a *cycle*. We can observe that any sequence of u-nontransient edges (*i.e.* a run of the product automaton, or its finite fragment) is of one of the following shapes: a cycle (called a *u-nontransient cycle*), a lasso shape (a simple path leading to a u-nontransient cycle), or a simple path ending at a state where the input u is not available at all. Informally, the existence of a stuttering self loop in a state l under input u in $\mathcal{T}_c$ means that this self loop cannot be followed infinitely many times in a row. Similarly, any u-nontransient cycle in the product graph cannot be followed infinitely many times in a row without leaving it. This leads us to the new computation of protagonist's and adversary's attractor.

Definition 7: The *protagonist's direct attractor* of S', denoted by $\mathsf{A}_P^1(S')$, is the set of all states $s \in S_{\mathcal{P}}$, such that there exists an input u satisfying

(1) $\delta_{\mathcal{P}}(s,u) \subseteq S'$, or
(2) s lies on a u-nontransient cycle, such that each state s' of the cycle satisfies that $s'' \in S'$ for all transient edges (s',u,s'')

In other words, the protagonist can enforce a visit to S' not only by entering S', but also by following a u-nontransient cycle finitely many times and eventually leaving it to S'.

Definition 8: The *adversary's direct attractor* of S', denoted by $\mathsf{A}_S^1(S')$, is the set of all states $s \in S_{\mathcal{P}}$, such that for each input u there exists a state s' such that

(1) $s' \in \delta_{\mathcal{P}}(s,u) \cap S'$, and
(2) s' does not lie on a u-nontransient cycle

In other words, the adversary cannot enforce a visit to S' via an edge of a u-nontransient cycle. This edge can be taken only finitely many times in row and eventually different edge under input u has to be chosen.

The protagonist's attractor of S' is then computed iteratively as the converging sequence $\mathsf{A}_{P0}^*(S') \subseteq \mathsf{A}_{P1}^*(S') \subseteq \mathsf{A}_{P2}^*(S') \subseteq \ldots$, where $\mathsf{A}_{P0}^*(S') = S'$ and $\mathsf{A}_{Pi+1}^*(S') = \mathsf{A}_P^1(\mathsf{A}_{Pi}^*(S'))$. The adversary's attractor is computed analogously and the rest of the construction of the control strategy for $\mathcal{T}_e$ remains unchanged. The additional computation described above allows us to consider information about stuttering and reduce the conservativeness of our method.

VII. Complexity and Conservativeness

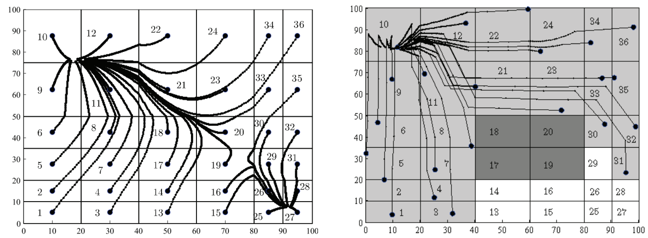
The complexity of the overall method is the cumulative complexity of (1) the construction of the control transition system $\mathcal{T}_c$ and (2) the generation of a control strategy for $\mathcal{T}_c$. The computation of $\mathcal{T}_c$ described in [25] involved enumerating all subsets of L at any element of L, which gives $O(|L| \cdot 2^{|L|})$ iterations, although this was significantly reduced through additional optimizations. At each iteration, polyhedral operations were performed, which scale exponentially with N, the size of the continuous state space. The characterization of stuttering inputs described in this paper checks each element from Σ_c through polyhedral operations. In [25] we also described a procedure for reducing the size of $\mathcal{T}_c$ by eliminating "more nondeterministic" transitions and showed that no solutions are lost in this process. This reduction can also be applied for the extended method described in this paper, but transitions under inputs in the sets Σ_c^{ls} and Σ_c^{lu} must be considered separately.

The overall complexity of the control strategy synthesis by Horn is $O(k!n^k)$, where n is the size of the product automaton and k is the number of pairs in the Rabin condition of the product automaton. The modifications we made in order to adapt the algorithm to deal with stuttering behavior do not change the overall complexity. Note that, in general, Rabin games are NP-complete, so the exponential complexity with respect to k is not surprising. However, LTL formulas are usually translated into Rabin automata with very few tuples in their acceptance condition.

Our solution to Problem 1 is obviously conservative. Note that the only source of conservativeness is the construction of the control transition system $\mathcal{T}_c$ - the solution to the LTL control problem for $\mathcal{T}_c$ is complete.

VIII. Implementation and Case Study

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 [17]. The tool takes as input a PWA system (as defined in



A B

Fig. 2: A) Trajectories of the uncontrolled PWA system go towards one of two possible stable equilibria located in regions X_{10} and X_{27}. **B)** Trajectories of the closed loop PWA system originating anywhere in the satisfying region (light gray) satisfy the specification and eventually reach and remain in region X_{10}, while avoiding regions X_{17}, X_{18}, X_{19}, and X_{20} (dark gray).

Eqn. (1)) and an LTL formula and produces a set of satisfying initial regions and a feedback control strategy for the system. The tool is available at http://hyness.bu.edu/software.

We analyzed a planar PWA system ($N = M = 2$ in Eqn. (1)) with 36 polytopes (Fig. 2-A, where only the labels of the polytopes are shown). The exact dynamics of the system are omitted due to space constraints. Trajectories of the uncontrolled system (shown in Fig. 2-A) go towards one of two possible stable equilibria located in regions X_{10} and X_{27}. We are interested in finding a control strategy, satisfying specification "*eventually visit region* X_{10} *and remain there forever and always avoid* X_{17}, X_{18}, X_{19}, *and* X_{20}", which can be written as $\phi = \Diamond\Box 10 \wedge \Box\neg(17 \vee 18 \vee 19 \vee 20)$. Formula ϕ can be translated into a deterministic Rabin automaton containing one tuple in the Rabin acceptance condition.

A control transition system $\mathcal{T}_c$ with 36 states was constructed. Out of the total 396 nonempty input regions found (denoted by $U_l^{L'}$ in Sec. IV), 274 were "large enough" (the radii of their inscribed spheres were larger than $\varepsilon = 0.05$) to be considered for a robust control strategy. The computation of $\mathcal{T}_c$ required 30 sec. and the construction of the control strategy an additional 1.5 min. on a 3.4 GHz, Intel Pentium 4 machine with 1GB of memory. The satisfying initial region identified by `conPAS2` is shown in light gray in Fig. 2-B. Starting from random initial conditions, trajectories of the closed loop system were simulated (Fig. 2-B), where at each step applied inputs were corrupted by noise. All simulated trajectories avoid the unsafe regions (shown in dark gray in Fig. 2-B) and satisfy the specification, thereby demonstrating the correctness and robustness of the control strategy. For this particular case study, satisfying control strategies can be identified only from region X_{10} unless the method is extended to deal with stuttering (Sec. VI).

IX. 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 polyhedral regions in its state space. Our approach consists of two main steps: (1) abstracting the original control system to a finite control system, and (2) generating a control strategy for the finite control system.

References

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

[2] 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. of PSB*, 2003, pp. 116–127.

[3] 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.

[4] 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 issue on grand challenges for robotics*, vol. 14, no. 1, pp. 61–71, 2007.

[5] E. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[6] J. M. Davoren, V. Coulthard, N. Markey, and T. Moor, "Non-deterministic temporal logics for general flow systems," in *Proc. of HSCC*, ser. LNCS, 2004, pp. 280–295.

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

[8] G. Frehse, S. K. Jha, and B. H. Krogh, "A counterexample-guided approach to parameter synthesis for linear hybrid automata," in *Proc. of HSCC*, ser. LNCS, 2008, pp. 187–200.

[9] 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.

[10] F. Horn, "Streett Games on Finite Graphs," in the *2nd Workshop on Games in Design and Verification (GDV)*, 2005.

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

[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. of HSCC*, ser. LNCS, 2005, pp. 354–369.

[13] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. of CDC*, 2008, pp. 2117–2122.

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

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

[16] H. Kress-Gazit, D. Conner, H. Choset, A. Rizzi, and G. Pappas, "Courteous cars," *IEEE Robotics and Automation Magazine*, vol. 15, pp. 30–38, March 2008.

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

[18] M.Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Proc. of HSCC*, ser. LNCS. Springer Berlin / Heidelberg, 2008, pp. 287–300.

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

[20] S. Safra, "On the complexity of omega-automata," in *Proc. of FOCS*, 1988, pp. 319–327.

[21] P. Tabuada and G. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[22] W. Thomas, "Infinite games and verification," in *Proc. of CAV*, 2002, pp. 58–64.

[23] J. Tůmová, B. Yordanov, C. Belta, I. Černá, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," Boston University, Tech. Rep. CISE 2010-IR-0014, 2010.

[24] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," in *Proc. of LICS*. Washington, DC, USA: IEEE Computer Society, 1986, pp. 332–344.

[25] B. Yordanov and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," in *Proc. of CDC*, 2009.

[26] B. Yordanov, C. Belta, and G. Batt, "Model checking discrete time piesewise affine systems: application to gene networks," in *Proc. of ECC*, 2007.