# A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications

Marius Kloetzer, *Student Member, IEEE*, and Calin Belta, *Member, IEEE*

***Abstract*—We consider the following problem: given a linear system and a linear temporal logic (LTL) formula over a set of linear predicates in its state variables, find a feedback control law with polyhedral bounds and a set of initial states so that all trajectories of the closed loop system satisfy the formula. Our solution to this problem consists of three main steps. First, we partition the state space in accordance with the predicates in the formula, and construct a transition system over the partition quotient, which captures our capability of designing controllers. Second, using a procedure resembling model checking, we determine runs of the transition system satisfying the formula. Third, we generate the control strategy. Illustrative examples are included.**

***Index Terms*—Control, model checking, temporal logic, transition systems.**

## I. Introduction

TEMPORAL LOGIC [1], [2] is the natural framework for specifying and verifying the correctness of digital circuits and computer programs. However, due to its resemblance to natural language, its expressivity, and the existence of off-the-shelf algorithms for model checking, temporal logic has the potential to impact several other areas of engineering. Analysis of systems with continuous dynamics based on qualitative simulations and temporal logic was proposed in [3] and [4]. In the control-theoretic community, a framework for specifying and controlling the behavior of a discrete linear system has been developed in [5]. In [6], the authors consider the problem of robustly controlling hybrid systems based on temporal logic specifications. The use of temporal logic for task specification and controller synthesis in mobile robotics has been advocated as far back as [7], and recent results include [8]–[12]. In the area of systems biology, the qualitative behavior of genetic circuits can be expressed in temporal logic, and model checking can be used for analysis, as suggested in [13] and [14]. Besides temporal logic, regular expressions represent another formalism for describing desired behaviors of real systems, as in [15]. The main difference is that regular expressions specify finite behaviors, whereas the temporal logic that we use specifies infinite ones.

Manuscript received April 19, 2006; revised April 11, 2007. Recommended by Associate Editor J. Hespanha. This work was supported by the National Science Foundation under CAREER Grant 0447721 and Grant 0410514. Preliminary results of this work were presented at the 9th International Workshop on Hybrid Systems: Computation and Control, Santa Barbara, CA, March 2006.

The authors are with the Center for Information and Systems Engineering, Boston University, Boston, MA 02446 USA (e-mail: kmarius@bu.edu; 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.2007.914952

We consider the following problem: *given a linear system $\dot{x} = Ax + b + Bu$ with polyhedral control constraints $U$, and given a specification in terms of an arbitrary linear temporal logic $(LTL)$ formula $\phi$ over an arbitrary set of linear predicates in $x$, find initial states and a feedback control strategy so that the corresponding trajectories of the closed loop system satisfy formula $\phi$, while staying inside a given full-dimensional polytope $P$.*

Our approach in solving the aforementioned problem can be summarized in the following three steps. In the first step, we construct a finite-state "generator" transition system $T_g$. Its states are the equivalence classes produced by the feasible full-dimensional subpolytopes of $P$ determined by the linear predicates appearing in formula $\phi$. The transitions of $T_g$ are determined by adjacency of subpolytopes and existence of feedback controllers, making such subpolytopes invariant or driving all states in a subpolytope to an adjacent subpolytope through a common facet [16]. In the second step, we produce runs of $T_g$ that satisfy formula $\phi$. This is in essence a model checking problem, and we use standard tools based on Büchi automata [17]. In the third step, we construct a feedback "control strategy," which leads to a closed loop hybrid system, whose continuous trajectories satisfy formula $\phi$. We implemented our approach as a user friendly software package LTLCon [18] under Matlab.

*1) Related work and contribution of the paper:* In order to extend temporal logic techniques from purely discrete systems to continuous systems, two approaches are possible. First, a careful treatment of the semantics of temporal logic formulas in models with continuous or hybrid dynamics [19] can be performed. Second, finite quotients with respect to meaningful equivalence relations can be constructed. Such equivalence relations include language equivalences (preserving properties specified in linear temporal logic) and bisimulation relations (preserving specifications in both linear and branching time logic). The first success in this direction was the work on timed automata reported in [20], followed by multirate automata [21] and rectangular hybrid automata [22]. Other classes of systems for which finite bisimulation quotients exist are identified in [23]. The interested reader is referred to [24] for an excellent review of all these works. Linear dynamics are studied in [25], while nonlinear systems are considered in [26] and [27]. Quotients that only simulate a continuous or hybrid system and can be used for conservative analysis are developed in [28] and [29].

This paper is inspired from [5] and [10]. The problem of controller synthesis from LTL specifications for discrete-time continuous-space linear systems with semilinear partitions are considered in [5], where it is shown that finite bisimulations exist for controllable systems with properly chosen observables.

The focus in [5] is on the existence and computability. Specifically, it is shown that the iterative (partitioning) bisimulation algorithm [23] terminates and each step is computable. However, no computational formulas for the controllers are provided. Another contribution of [5] is setting up of the framework for producing runs of the finite quotient satisfying an LTL formula. This framework is further refined in [10], where the authors study the problem of controlling a planar robot in a polygon, so that its trajectory satisfies an LTL$_{-X}$ formula. In [10], it is assumed that a triangulation of the polygon is given, and vector fields are assigned in each triangle so that the produced trajectories satisfy a formula over the triangles. For the construction of vector fields, the authors use the algorithms developed in [30].

This paper extends the results of [5] and [10] in several ways. First, we consider continuous-time systems as opposed to discrete-time systems in [5]. Second, based on results on controlling a linear system to a facet of a polytope from [16], and an invariance theorem stated and proved in this paper, we provide a fully computational and algorithmic approach to controller design consisting of polyhedral operations and searches on graphs only. Third, as opposed to [5], we can guarantee arbitrary polyhedral control bounds. Fourth, we extend the results [10] by approaching arbitrary-dimensional problems and considering systems with (linear) drift. The feasibility of the partition induced by the predicates in the formula and the construction of the partition quotient is fully automated in our framework, rather than assuming a given triangulation. Finally, we provide a tighter connection between the continuous and the discrete part of the problem in two ways. First, the transitions of the discrete quotient capture the controllability properties of the continuous system. Second, the runs of the discrete system are shown to be of a particular form, which is implementable by the continuous system.

*2) Organization of the paper:* The remainder of the paper is organized as follows. Section II provides some preliminaries necessary throughout the paper. The problem is formulated in Section III. The construction of the generator transition system is presented in Section IV, while its runs satisfying the formula are found in Section V. The control strategy providing a solution to the main problem is presented in Section VI. The conservativeness and complexity of our approach are discussed in Section VII, while implementation notes and simulation results are given in Section VIII. We conclude with final remarks and directions for future work in Section IX.

## II. Preliminaries

### A. Polytopes

Let $N \in \mathbb{N}$ and consider the $N$-dimensional Euclidean space $\mathbb{R}^N$. A full-dimensional *polytope* $P$ is defined as the convex hull of at least $N+1$ affinely independent points in $\mathbb{R}^N$. A set of $M \geq N+1$ points $v_1, \ldots, v_M \in \mathbb{R}^N$ whose convex hull gives $P$ and with the property that $v_i$, $i = 1, \ldots, M$ is not contained in the convex hull of $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_M$ is called the set of *vertices* of $P$. A polytope is completely described by its set of vertices:

$$P = conv(v_1, \ldots, v_M), \tag{1}$$

where $conv$ denotes the convex hull. Alternatively, $P$ can be described as the intersection of at least $N+1$ closed half spaces. In other words, there exist a $K \geq N+1$ and $a_i \in \mathbb{R}^N$, $b_i \in \mathbb{R}$, $i = 1, \ldots, K$, such that

$$P = \left\{x \in \mathbb{R}^N \mid a_i^T x + b_i \leq 0, \;\; i = 1, \ldots, K\right\}. \tag{2}$$

Forms (1) and (2) are referred to as V- and H-representations of the polytope, respectively. Given a full-dimensional polytope $P$, there exist algorithms for translation from representation (1) to representation (2) [31], [32]. A *face* of $P$ is the intersection of $P$ with one or several of its supporting hyperplanes. If the dimension of the intersection is $p$ (with $0 \leq p < N$), then the face is called a $p$-face. An $(N-1)$-face obtained by intersecting $P$ with one of its supporting hyperplanes is called a *facet*. The vertices of $P$ are 0-faces. We denote by $int(P)$ the set of points of $P$, which are not on its facets, i.e., the region in $\mathbb{R}^N$ obtained if the inequalities in (2) were strict. If $F$ is a facet of $P$, $int(F)$ is defined analogously with the observation that $F$ is a full-dimensional polytope in $\mathbb{R}^{N-1}$.

A full-dimensional polytope with $N+1$ vertices (and $N+1$ facets) is called a full-dimensional *simplex*. Arbitrary full-dimensional polytopes can be *triangulized* [33]. In other words, for any full-dimensional polytope $P$, there exist full-dimensional simplices $S_1, \ldots, S_L$ such that: 1) $P = \bigcup_{i=1}^{L} S_i$; 2) $S_i \bigcap S_j$ is either empty or a common face of $S_i$ and $S_j$, for all $i, j = 1, \ldots, L$, $i \neq j$; and 3) the set of vertices of simplex $S_i$ is a subset of $\{v_1, \ldots, v_M\}$, for all $i = 1, \ldots, L$.

The interested reader is referred to [34] for more details on polytopes and to [33] and [35] for more information on triangulations. Available packages for triangulations and other polyhedral operations such as transformations between the two representations (1) and (2) are described in [36], [37], and [32].

### B. Transition Systems and Temporal Logic

*Definition 1:* A transition system is a tuple $T = (Q, Q_0, \rightarrow, \Pi, \models)$, where $Q$ is a set of states, $Q_0 \subseteq Q$ is a set of initial states, $\rightarrow \subseteq Q \times Q$ is a transition relation, $\Pi$ is a finite set of atomic propositions, and $\models \subseteq Q \times \Pi$ is a satisfaction relation.

In this paper, we assume that the transition system is *finite* ($Q$ is finite). For an arbitrary proposition $\pi \in \Pi$, we define $[\![\pi]\!] = \{q \in Q | q \models \pi\}$ as the set of all states satisfying it. Conversely, for an arbitrary state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \mid q \models \pi\}$, $\Pi_q \in 2^{\Pi}$, denote the set of all atomic propositions satisfied at $q$. A *trajectory* or *run* of $T$ starting from $q$ is an infinite sequence $r = r(1)r(2)r(3)\ldots$ with the property that $r(1) = q$, $r(i) \in Q$, and $(r(i), r(i+1)) \in \rightarrow$, for all $i \geq 1$. A trajectory $r = r(1)r(2)r(3)\ldots$ defines a *word* $w = w(1)w(2)w(3)\ldots$, where $w(i) = \Pi_{r(i)}$.

In the rest of this section, we give a brief review of a propositional linear temporal logic known as LTL$_{-X}$ [1].

*Definition 2 (Syntax of LTL$_{-X}$ formulas):* A linear temporal logic LTL$_{-X}$ formula over $\Pi$ is recursively defined as follows:

1) every atomic proposition $\pi_i$ is a formula; and
2) if $\phi_1$ and $\phi_2$ are formulas, then $\phi_1 \vee \phi_2$, $\neg\phi_1$, $\phi_1 \mathcal{U} \phi_2$ are also formulas.

The semantics of LTL$_{-X}$ formulas are given over words of transition system $T$.

*Definition 3* (*Semantics of LTL$_{-X}$ formulas*)*:* The satisfaction of formula $\phi$ at position $i \in \mathbb{N}$ of word $w$, denoted by $w(i) \models \phi$, is defined recursively as follows:

1) $w(i) \models \pi$, if $\pi \in w(i)$;
2) $w(i) \models \neg\phi$, if $w(i) \not\models \phi$ (where $\not\models$ denotes the negation of $\models$);
3) $w(i) \models \phi_1 \vee \phi_2$, if $w(i) \models \phi_1$ or $w(i) \models \phi_2$;
4) $w(i) \models \phi_1 \mathcal{U} \phi_2$, if there exist a $j \geq i$ such that $w(j) \models \phi_2$ and for all $i \leq k < j$ we have $w(k) \models \phi_1$.

A word $w$ satisfies an LTL$_{-X}$ formula $\phi$, written as $w \models \phi$, if $w(1) \models \phi$.

The symbols $\neg$ and $\vee$ stand for negation and disjunction. The Boolean constants $\top$ and $\bot$ are defined as $\top = \pi \vee \neg\pi$ and $\bot = \neg\top$. The other Boolean connectors $\wedge$ (conjunction), $\Rightarrow$ (implication), and $\Leftrightarrow$ (equivalence) are defined from $\neg$ and $\vee$ in the usual way. The *temporal operator* $\mathcal{U}$ is called the *until* operator. Formula $\phi_1 \mathcal{U} \phi_2$ intuitively means that (over a word) $\phi_2$ will eventually become true, and $\phi_1$ is true until this happens. Two useful additional temporal operators, "eventually" and "always," can be defined as $\Diamond\phi = \top\mathcal{U}\phi$ and $\Box\phi = \neg\Diamond\neg\phi$, respectively. Formula $\Diamond\phi$ means that $\phi$ becomes eventually true, whereas $\Box\phi$ indicates that $\phi$ is true at all positions of $w$. More expressivenesses can be achieved by combining the temporal operators. Examples include $\Box\Diamond\phi$ ($\phi$ is true infinitely often) and $\Diamond\Box\phi$ ($\phi$ becomes eventually true and stays true forever).

The LTL [1], [2], the most used propositional linear temporal logic, is richer than the LTL$_{-X}$ in the sense that it allows for an additional temporal operator $\circ$, which is called the "next" operator. Formally, the syntax of LTL is obtained by adding "$\circ\phi_1$" to Definition 2, and its semantics is defined by adding "$w(i) \models \circ\phi$ if $w(i+1) \models \phi$" to Definition 3. A careful examination of the LTL and LTL$_{-X}$ semantics shows that the increased expressivity of LTL is manifested only over words with a finite number of repetitions of a symbol. Consider, for example, the words $w = \pi_1\pi_2\pi_3\ldots$, $w' = \pi_1\pi_2\pi_2\pi_3\ldots$, and $w = \pi_1\pi_2\pi_2\pi_2\ldots$. Then, in LTL, while all $w$, $w'$, $w''$ satisfy formula $\circ\pi_2$, we can distinguish between $w$ and $w'$ with formula $\circ\circ\pi_2$, which is true for $w'$ and false for $w$. On the other hand, $w''$ satisfies $\circ\Box\pi_2$, which is false for both $w$ and $w'$. In the LTL$_{-X}$, we can distinguish between $w''$ and $w$ or $w'$, because formula $\Diamond\Box\pi_2$ is true for $w''$ and false for both $w$ and $w'$. However, we cannot distinguish between $w$ and $w'$, which would require the $\circ$ operator. Our choice of LTL$_{-X}$ over LTL is motivated by our definition of the satisfaction of a formula by a continuous trajectory, and by our approach to finding runs. Specifically, as it will become clear in Section III, a word corresponding to a continuous trajectory will never have a finite number of successive repetitions of a symbol.

## III. Problem Formulation and Approach

Consider the following affine control system in a full-dimensional polytope $P$ in $\mathbb{R}^N$

$$\dot{x} = Ax + b + Bu, \quad x \in P,\ u \in U \subset \mathbb{R}^m \tag{3}$$

where $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times m}$, $b \in \mathbb{R}^N$, and $U$ is a given polyhedral subset of $\mathbb{R}^m$ capturing control constraints. Let $\Pi$ be a set of atomic propositions given as arbitrary strict linear inequalities in $\mathbb{R}^N$. Formally

$$\Pi = \{\pi_i \mid i = 1, \ldots, n\} \tag{4}$$

where each proposition $\pi_i$, $i = 1, \ldots, n$, denotes an open half-space of $\mathbb{R}^N$:

$$[\![\pi_i]\!] = \left\{x \in \mathbb{R}^N \mid c_i^T x + d_i < 0\right\}, \tag{5}$$

with $c_i \in \mathbb{R}^N$ and $d_i \in \mathbb{R}$.

The polytope $P$ can be seen as a region of $\mathbb{R}^N$ capturing known physical bounds on the state of system (3), or as a region that is required to be an invariant for its trajectories. For example, for $N = 2$, $P$ can be a convex polygon giving the environment boundaries for a planar robot with kinematics given by (3). The predicates (5) describe other regions (properties) of interest. Note that, for technical reasons to become clear later, we only allow strict inequalities in (5). However, this assumption does not seem restrictive from an application point of view. If the predicates in $\Pi$ model sensor information, it is unrealistic to check for the attainment of a specific value due to sensor noise. Moreover, if a specific value is of interest, it can be included in the interior of a polyhedron given by other predicates.

In this paper, we consider the following problem:

*Problem 1:* For an arbitrary LTL$_{-X}$ formula $\phi$ over $\Pi$, find a set of initial states and a feedback control strategy for system (3) so that all trajectories of the corresponding closed-loop system satisfy $\phi$, while always staying inside $P$.

To fully specify Problem 1, we need to define the satisfaction of an LTL$_{-X}$ formula $\phi$ on $\Pi$ by a trajectory of (3), which can be seen as a continuous curve $a : [0, \infty) \to \mathbb{R}^N$. This curve can, in general, be nonsmooth and can have self-intersections. For each $\Theta \in 2^\Pi$, we define $[\![\Theta]\!]$ as being the set of states in $\mathbb{R}^N$ satisfying all and only propositions $\pi \in \Theta$

$$[\![\Theta]\!] = \bigcap_{\pi\in\Theta} [\![\pi]\!] \setminus \bigcup_{\pi\in\Pi\setminus\Theta} [\![\pi]\!]. \tag{6}$$

*Definition 4:* The *word* corresponding to trajectory $a$ is the sequence $w_a = w_a(1)\, w_a(2) w_a(3)\ldots$, $w_a(k) \in 2^\Pi$, $k \geq 1$, generated such that the following rules are satisfied for all $\tau \geq 0$ and $k \in \mathbb{N}$, $k \geq 1$:

1) $a(0) \in [\![w_a(1)]\!]$;
2) if $a(\tau) \in [\![w_a(k)]\!]$ and $w_a(k) \neq w_a(k+1)$, then there exist $\tau' > \tau$ such that: a) $a(\tau') \in [\![w_a(k+1)]\!]$, b) $a(t) \notin [\![\pi]\!]$, $\forall t \in [\tau, \tau']$, $\forall \pi \in \Pi \setminus (w_a(k) \cup w_a(k+1))$, and c) $c_i^T a(t') + d_i \neq 0$, for all $i \in \{1, \ldots, n\}$ and $t' \in \{\tau, \tau'\}$;
3) if $a(\tau) \in [\![w_a(k)]\!]$ and $w_a(k) = w_a(k+1)$, then $a(t) \in [\![w_a(k)]\!]$, $\forall t \geq \tau$ (i.e., the region $[\![w_a(k)]\!]$ is a "sink" for trajectory $a$).

A careful examination of Definition 4 shows that the word produced by a continuous trajectory is exactly the sequence of sets of propositions satisfied by it as time evolves. Some illustrative examples of words defined by continuous trajectories are given in Fig. 1, where $\pi_i$, $i = 1, \ldots, 6$ are open half-spaces and $\Pi_1 = \{\pi_1, \pi_3\}, \Pi_2 = \{\pi_1, \pi_3, \pi_5\}, \Pi_3 = \{\pi_1, \pi_2, \pi_3, \pi_5\}$,

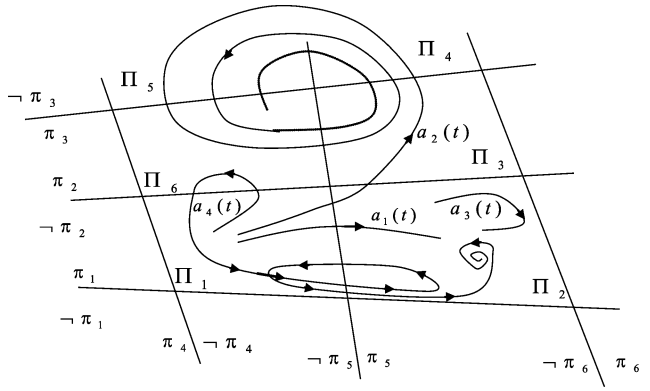



Fig. 1. Examples of continuous trajectories for Definition 4.

$\Pi_4 = \{\pi_1, \pi_2, \pi_5\}$, $\Pi_5 = \{\pi_1, \pi_2\}$, $\Pi_6 = \{\pi_1, \pi_2, \pi_3\}$. Continuous trajectory $a_1$ starts from the region where the predicates in $\Pi_1$ are true and converges to a point in the region where the predicates in $\Pi_2$ are true. By the earlier definition, the word $w_{a_1}$ is $\Pi_1\Pi_2\Pi_2\ldots$. Trajectory $a_2$ starts from $\Pi_1$ and loops infinitely, as shown in the figure. The corresponding word $w_{a_2}$ will be $\Pi_1\Pi_2\Pi_3\Pi_4\Pi_5\Pi_6\Pi_3\Pi_4\Pi_5\Pi_6\ldots$. For trajectory $a_3$ originating in $\Pi_2$ and converging inside $\Pi_2$, the word $w_{a_3}$ is $\Pi_2\Pi_2\Pi_2\ldots$. Finally, trajectory $a_4$, which has a self-intersection, generates the word $w_{a_4} = \Pi_1\Pi_6\Pi_1\Pi_2\Pi_1\Pi_2\Pi_2\ldots$.

*Remark 1:* On the well posedness of Definition 4, first note that our assumption that trajectories of system (3) always stay inside $P$ implies that the generated words have infinite length, so the problem of satisfaction of an LTL$_{-X}$ by such a word is well-posed. Second, since the predicates in (4) are given by strict linear inequalities, Definition 4 makes sense only if $c_i^T a(0) + d_i \neq 0$ and $c_i^T \bar{a} + d_i \neq 0$, where $\bar{a} = \lim_{t\to\infty} a(t)$ (if it exists), for all $i = 1, \ldots, n$. Third, Definition 4 is a proper characterization of satisfaction of sets of predicates from $\Pi$ by $a(t)$ as time evolves only if there does not exist $t_1 < t_2$ and $i = 1, \ldots, n$, such that $c_i^T a(t) + d_i = 0$, for all $t \in (t_1, t_2)$. All these three requirements are guaranteed by the way we design controllers, as it will become clear in Sections IV-A and VI.

*Remark 2:* According to Definition 4, the word $w_a$ produced by a trajectory $a(t)$ does not contain a finite number of successive repetitions of a symbol, which suggests using LTL without the "next" operator, as stated in Section II-B.

*Definition 5:* A trajectory $a : [0, \infty) \to \mathbb{R}^N$ of (3) satisfies LTL$_{-X}$ formula $\phi$, written as $a \models \phi$, if and only if $w_a \models \phi$, where $w_a$ is the word generated in accordance with Definition 4.

## IV. Generator Transition System

In this section, we define the generator transition system $T_g$, which captures our ability to design feedback controllers for (3). We start by stating a theorem from [16], which gives a sufficient condition for the existence of an affine feedback controller driving all initial states of a linear system in a polytope through a facet in finite time. Using similar ideas, we state and prove a necessary and sufficient condition for invariance in a polytope. We then use these characterizations to construct $T_g$.

### A. Control of Affine Systems in Polytopes

Consider a full-dimensional polytope $P$ in $\mathbb{R}^N$ with vertices $v_1, \ldots, v_M$, $M \geq N+1$. Let $F_1, \ldots, F_K$ denote the facets of $P$ with normal vectors $n_1, \ldots, n_K$ pointing out of the polytope $P$. For $i = 1, \ldots, K$, let $V_i \subset \{1, \ldots, M\}$ be the set of indexes of vertices belonging to facet $F_i$. For $j = 1, \ldots, M$, let $W_j \subset \{1, \ldots, K\}$ be the set of indexes of all facets containing vertex $v_j$.

*Lemma 1 (Lemma 4.6 from: [16]):* There exists a continuous function $\lambda : P \to [0,1]^M$ with $\sum_{j=1}^M \lambda_j(x) = 1$, such that, for all $x \in P$, $x = \sum_{j=1}^M \lambda_j(x) v_j$.

*Theorem 1 (Theorem 4.7 plus Remark 4.8 from [16]):* Consider control system (3) defined on the full-dimensional polytope $P$. Assume that there exist $u_1, \ldots, u_M \in U$ such that:

1) $\forall j \in V_1$ :
   a) $n_1^T(Av_j + Bu_j + b) > 0$
   b) $\forall i \in W_j \setminus \{1\}$: $n_i^T(Av_j + Bu_j + b) \leq 0$;
2) $\forall j \in \{1, \ldots, M\} \setminus V_1$ :
   a) $\forall i \in W_j$: $n_i^T(Av_j + Bu_j + b) \leq 0$
   b) $n_1^T(Av_j + Bu_j + b) > 0$.

Then, there exists a continuous feedback controller $u$: $P \to U$ with the property that for any initial state $x(0) \in P$, there exist a $T_0 > 0$ such that: 1) $\forall t \in [0, T_0]$: $x(t) \in P$; 2) $x(T_0) \in F_1$; and 3) $n_1^T \dot{x}(T_0) > 0$.

In other words, Theorem 1 states that if linear inequalities (1)(a),(b) and (2)(a),(b) are satisfied by some $u_1, \ldots, u_M \in U$, then a continuous feedback controller driving all initial states from $P$ out of $P$ through facet $F_1$ in finite time exists (condition 3 means that the velocity on the exit facet $F_1$ is oriented outside the facet).

*Theorem 2:* Consider control system (3) defined on the full-dimensional polytope $P$. There exists a continuous feedback controller $u : P \to U$ that makes $P$ an invariant for (3) if and only if there exist $u_1, \ldots, u_M \in U$ such that

$$\forall j \in \{1, \ldots, M\}, \forall i \in W_j : \; n_i^T(Av_j + Bu_j + b) \leq 0.$$

*Proof:* For necessity, assume that $u(x)$ is a continuous feedback controller so that $P$ is an invariant for $\dot{x} = Ax + Bu(x) + b$. Let $u_j = u(v_j)$, $j = 1, \ldots, M$. Assume by contradiction that there exist $j = 1, \ldots, M$ and $i \in W_j$ such that $n_i^T(Av_j + Bu_j + b) > 0$. Then, the closed loop system $\dot{x} = Ax + Bu(x) + b$ starting at $v_j$ will leave the polytope. In fact, by continuity of the vector field, there will exist a whole set of points on $F_i$ leaving the polytope, which contradicts the assumption.

For sufficiency, by applying Lemma 1 to an arbitrary facet $F_i$ (which can be seen as a full-dimensional polytope in $\mathbb{R}^{N-1}$), any $x \in F_i$ can be written as $x = \sum_{j \in V_i} \lambda_j(x) v_j$. Let $u(x) = \sum_{j=1}^M \lambda_j(x) u_j$. Since $u_j \in U$, $j = 1, \ldots, M$, of course $u(x) \in U$, for all $x \in P$. Also, $Ax + Bu(x) + b = \sum_{j=1}^M \lambda_j(x)(Av_j + Bu_j + b)$, for all $x \in P$. For any $i \in \{1, \ldots, K\}$, we have $n_i^T(Ax + Bu(x) + b) = \sum_{j \in V_i} \lambda_j(x)(n_i^T(Av_j + Bu_j + b)) \leq 0$ by hypothesis. In other words, the velocity of the closed loop system is oriented "inwards" for all facets, and therefore, $P$ is an invariant for its trajectories. ∎

For both Theorems 1 and 2, given the values $u_1, \ldots, u_M$ at the vertices, the construction of a continuous controller everywhere in $P$ starts with a triangulation $S_1, \ldots, S_L$ of $P$. Let $v_1^i, \ldots, v_{N+1}^i \in \{v_1, \ldots, v_M\}$ be the vertices of the full-dimensional simplex $S_i$, $i = 1, \ldots, L$, and $u_1^i, \ldots, u_{N+1}^i \in \{u_1, \ldots, u_M\}$ be the corresponding control values. Then, everywhere in $P$, the feedback control is given by

$$u(x) = u^i(x) \text{ if } x \in S_i, \quad i = 1, \ldots, L \tag{7}$$

where the control in each simplex is given by [38]

$$u^i(x) = \begin{bmatrix} u_1^i & \cdots & u_{N+1}^i \end{bmatrix} \begin{bmatrix} v_1^i & \cdots & v_{N+1}^i \\ 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad i = 1, \ldots, L. \tag{8}$$

Note that the controller given by (7) is well defined. It is obvious that the controller is well defined when (7) is restricted to the interior of the simplices, since the intersection of all such interiors is empty. The only problem that might appear is on the common facets. However, recall that an affine function defined on $\mathbb{R}^N$ is uniquely determined by its values at the vertices of a full-dimensional simplex, and the restriction of the function to the simplex is a unique convex combination of these values [16], [38]. Moreover, a facet of a full-dimensional simplex in $\mathbb{R}^N$ is a full-dimensional simplex in $\mathbb{R}^{N-1}$. It follows that, given a pair of adjacent simplices $S_i$ and $S_j$, $u^i(x) = u^j(x)$ everywhere on the common facet of $S_i$ and $S_j$. Therefore, (7) is well defined, and the affine feedback controller is continuous everywhere in $P$. Moreover, $u(x)$, constructed using (7), is always a convex combination of the values $u_1, \ldots, u_M$. This guarantees that $u(x) \in U$ everywhere in $P$ if and only if $u_j \in U$, for all $j = 1, \ldots, M$.

If inequalities (1)(b) and (2)(a) from Theorem 1 are satisfied strictly, then it can be seen that, for all $i = 2, \ldots, K$ and all $j \in V_i$, $n_i^T (Av_j + Bu_j + b) < 0$. Since with $u$ constructed using (7) and (8), the restriction of $n_i^T (Ax + Bu + b)$ to $F_i$ is a convex combination of $n_i^T (Av_j + Bu_j + b)$, $j \in V_i$, it follows that $n_i^T (Ax + Bu + b) < 0$ everywhere in $F_i$. We conclude that, if the system starts in $int(P)$, it will never reach $F_i$. Moreover, if it starts in $F_i$, it will instantaneously penetrate in $int(P)$. Similar reasoning applies to the case when the inequalities of Theorem 2 are strict, leading to the following two corollaries:

*Corollary 1:* If inequalities (1)(b) and (2)(a) from Theorem 1 are satisfied strictly, the continuous controller constructed in accordance with (7) and (8) produces trajectories that satisfy $x(t) \in int(P)$, for all $t \in (0, T_0)$ and $x(T_0) \in int(F_1)$.

*Corollary 2:* If the inequalities in Theorem 2 are satisfied strictly, then $int(P)$ is an invariant for system (3) with controls given by (7) and (8).

### B. Construction of the Generator Transition System

Assume that the polytope $P$ is given in the inequality form (2). We denote by $\mathcal{M}$, $1 \leq \mathcal{M} \leq 2^n$ the number of feasible sets of the form $\bigwedge_{i=1}^n ((-1)^{j_i} (c_i^T x + d_i) < 0) \bigwedge_{l=1}^K (a_l^T x + b_l < 0)$, where $j_1, \ldots, j_n \in \{0, 1\}$ (each of these sets is the interior of a full-dimensional polytope included in $P$, which corresponds to a feasible combination of all predicates from $\Pi$ inside $P$). To each of them, we attach a symbol $q_i$, $i = 1, \ldots, \mathcal{M}$. Let $\mathcal{P}$ denote the set of all such symbols $\mathcal{P} = \{q_i \,|\, i = 1, \ldots, \mathcal{M}\}$. Let $h : P_{N_-} \to \mathcal{P}$ be the quotient map corresponding to these nonempty sets, where $P_{N_-} = int(P) \setminus \bigcup_{i=1}^n \{x \in \mathbb{R}^n \,|\, c_i^T x + d_i = 0\}$. We also use the notations $h^{-1}(q)$ and $h^{-1}(h(x))$ to denote the set of all points in $P_{N_-}$ with quotient $q$ and the set of all points in $P_{N_-}$ in the same equivalence class with $x$, respectively. Let $\overline{h^{-1}(q)}$ denote the closure of $h^{-1}(q)$. Note that $\overline{h^{-1}(q)}$, $q \in \mathcal{P}$ are full-dimensional subpolytopes of $P$. It is easy to see that $h^{-1}(q_i) \bigcap h^{-1}(q_j) = \emptyset$ for all $i, j = 1, \ldots, \mathcal{M}$, $i \neq j$ and $\bigcup_{i=1}^{\mathcal{M}} \overline{h^{-1}(q_i)} = P$.

*Definition 6:* The transition system $T_g = (Q_g, Q_{g0}, \rightarrow_g, \Pi_g, \models_g)$ is defined by:

1) $Q_g = Q_{g0} = \mathcal{P}$;
2) for all $i = 1, \ldots, \mathcal{M}$, $(q_i, q_i) \in \rightarrow_g$, if there exists a feedback controller $u_{q_i q_i} : \overline{h^{-1}(q_i)} \to U$ for the polytope $\overline{h^{-1}(q_i)}$, making $h^{-1}(q_i)$ an invariant for the trajectories of (3), as in Corollary 2 of Theorem 2;
3) for all $i, j = 1, \ldots, \mathcal{M}$, $i \neq j$, $(q_i, q_j) \in \rightarrow_g$ if $\overline{h^{-1}(q_i)}$ and $\overline{h^{-1}(q_j)}$ share a facet, and there exists a feedback controller $u_{q_i q_j} : \overline{h^{-1}(q_i)} \to U$ for the polytope $\overline{h^{-1}(q_i)}$ with exit facet $\overline{h^{-1}(q_i)} \cap \overline{h^{-1}(q_j)}$, as in Corollary 1 of Theorem 1,
4) $\Pi_g = \Pi$, with $\Pi$, as defined in (4);
5) $q \models_g \pi_i \in \Pi$ if $\exists x \in h^{-1}(q)$, so that $c_i^T x + d_i < 0$.

On the computation of the transition system $T_g$, (i.e., checking the existence of affine controllers $u_{q_i q_i}$ and $u_{q_i q_j}$), it is important to note that it only consists of checking the nonemptiness of polyhedral sets (since $U$ is polyhedral), for which there exists several powerful algorithms.

## V. Determining Trajectories of the Generator Transition System

For $T_g$ constructed as shown in the previous section, we need to find runs satisfying an arbitrary LTL$_{-X}$ formula over its predicates. In this section, we develop a general algorithm, which takes as input an arbitrary transition system $T = (Q, Q_0, \rightarrow, \Pi, \models)$ (Definition 1) and an arbitrary LTL$_{-X}$ formula $\phi$ over $\Pi$. The algorithm returns the initial states of $T$ from which the formula can be satisfied, together with a satisfying run from each initial state. None of the produced runs will have a finite number of successive repetitions of a state, in accordance with the semantics of a continuous curve (Remark 2). In addition, the produced runs will be optimal in a sense to be defined later in this section.

The main steps of the algorithm are as follows. We start by translating $\phi$ into a Büchi automaton $\mathcal{B}_\phi$. Then, we take the product of $T$ with $\mathcal{B}_\phi$ to obtain a product automaton $\mathcal{A}_\phi$, whose accepted runs will only include trajectories of $T$ that satisfy formula $\phi$. We then find "short" runs of $\mathcal{A}_\phi$ that do not contain finite successive repetitions, and project back into runs of $T$.

Our approach is inspired from classical algorithms for LTL model checking. Such algorithms take as input a nonblocking

transition system $T$ (also called Kripke structure) and a formula $\phi$, and return the initial states of $T$ from which the formula is satisfied. For the initial states where the formula is not satisfied, a counterexample is returned. If there are blocking states in $T$, then the "stutter extension" rule [10], [39] can be applied when self-transitions are artificially added to blocking states.

Given the specific requirements of our problem, we could not use standard LTL model checkers. This is motivated by the fact that our transition system can have blocking states, and the use of the stutter extension rule might add spurious transitions that could not be implemented by a controller. In the rest of this section, we give the necessary definitions and describe the details of our algorithm.

*Definition 7 (Büchi automaton):* A Büchi automaton is a tuple $\mathcal{B} = (S, S_0, \Sigma, \rightarrow_{\mathcal{B}}, F)$, where:

1) $S$ is a finite set of states;
2) $S_0 \subseteq S$ is the set of initial states;
3) $\Sigma$ is the input alphabet;
4) $\rightarrow_{\mathcal{B}} \subseteq S \times \Sigma \times S$ is a nondeterministic transition relation;
5) $F \subseteq S$ is the set of accepting (final) states.

The semantics of a Büchi automaton is defined over infinite input words. Let $\omega = \omega_1\omega_2\omega_3 \ldots$ be an infinite input word of automaton $\mathcal{B}$, $\omega_i \in \Sigma$, $\forall i \in \mathbb{N} \setminus \{0\}$. We denote by $\mathcal{R}_B(\omega)$ the set of all initialized runs of $\mathcal{B}$ that can be generated by $\omega$

$$\mathcal{R}_{\mathcal{B}}(\omega) = \{r = s_1 s_2 s_3 \ldots | s_1 \in S_0,\ (s_i, \omega_i, s_{i+1}) \in \rightarrow_{\mathcal{B}},\ \forall i \in \mathbb{N} \setminus \{0\}\}. \quad (9)$$

*Definition 8 (Büchi acceptance):* A word $\omega$ is accepted by the Büchi automaton $\mathcal{B}$ (the word satisfies the automaton) if and only if $\exists r \in \mathcal{R}_B(\omega)$ so that $inf(r) \cap F \neq \emptyset$, where $inf(r)$ denotes the set of states appearing infinitely often in run $r$.

In [40], it was proved that, for any LTL formula $\phi$ over a set of atomic propositions $\Pi$, there exists a Büchi automaton $\mathcal{B}_\phi$ with input alphabet $\Sigma \subseteq 2^{\Pi}$ accepting *all and only* the infinite strings over $\Pi$ satisfying formula $\phi$. Translation algorithms were proposed in [40] and efficient implementations were developed in [41] and [17]. The interested reader is referred to [42] for a detailed tutorial on this matter. In this paper, we use the conversion algorithm described in [17] and its freely downloadable implementation, LTL2BA.

*Definition 9 (Product automaton):* The product automaton $\mathcal{A} = T \times \mathcal{B}$ between the transition system $T = (Q, Q_0, \rightarrow, \Pi, \models)$ and the Büchi automaton $\mathcal{B} = (S, S_0, \Sigma, \rightarrow_{\mathcal{B}}, F)$ with $\Sigma \subseteq 2^{\Pi}$ is defined as the tuple $\mathcal{A} = (S_{\mathcal{A}}, S_{\mathcal{A}0}, \rightarrow_{\mathcal{A}}, F_{\mathcal{A}})$, where:

1) $S_{\mathcal{A}} = (Q \cup \{q_0\}) \times S$ is the finite set of states;
2) $S_{\mathcal{A}0} = \{q_0\} \times S$ is the set of initial states;
3) $\rightarrow_{\mathcal{A}} \subseteq S_{\mathcal{A}} \times S_{\mathcal{A}}$ is the transition relation, defined as: $\{(q_i, s_j), (q_k, s_l)\} \in \rightarrow_{\mathcal{A}}$ if and only if $(q_i, q_k) \in (\rightarrow \cup (\{q_0\} \times Q_0))$ and $(s_j, \Pi_{q_k}, s_l) \in \rightarrow_{\mathcal{B}}$;
4) $F_{\mathcal{A}} = Q \times F$ is the set of accepting (final) states.

The acceptance condition of $\mathcal{A}$ is formulated similar to Definition 8, but with respect to runs instead of input words [10], [43]. Explicitly, a run $r_{\mathcal{A}}$ of $\mathcal{A}$ is accepted if it satisfies the transitions $\rightarrow_{\mathcal{A}}$ and $inf(r_{\mathcal{A}}) \cap F_{\mathcal{A}} \neq \emptyset$. The product automaton in Definition 9 can be regarded as a match between the states of $T$ and the transitions of $\mathcal{B}$, and therefore, $\mathcal{A}$ is sometimes referred to as the synchronous product [39]. The dummy state $q_0$, which has transitions to initial states of $T$ only, has been introduced for checking the possible satisfaction of a part of the formula from an initial state of $T$.

For any initialized run $r_{\mathcal{A}} = (q_0, s_{j1})(q_{i1}, s_{j2})(q_{i2}, s_{j3}) \ldots$ of automaton $\mathcal{A}$, we define the projection $\gamma_T(r_{\mathcal{A}}) = q_{i1}q_{i2} \ldots$, which maps $r_{\mathcal{A}}$ to the corresponding run of $T$. The following result is adapted from [10].

*Proposition 1:* If $\phi$ is an arbitrary LTL formula over $\Pi$ and $\mathcal{B}_\phi$ is a corresponding Büchi automaton, then the projection $\gamma_T(r_{\mathcal{A}_\phi})$ of any accepted run $r_{\mathcal{A}_\phi}$ of $\mathcal{A}_\phi = T \times \mathcal{B}_\phi$ is a run of $T$ satisfying LTL formula $\phi$.

Therefore, a run of $T$ satisfying specification $\phi$ exists if and only if $\mathcal{A}_\phi$ has an accepted run [10], [43]. This, in turn, is equivalent to the existence of a strongly connected component of the directed graph corresponding to $\mathcal{A}_\phi$ reachable from at least one initial state and containing at least one accepting state [39]. Among the accepted runs $r_{\mathcal{A}_\phi}$ of $\mathcal{A}_\phi$, we look for those with the particular structure of a *prefix* followed by infinitely many repetitions of a *suffix*. A prefix is a finite trajectory from an initial state to an accepting state (excluding the accepting state), while a suffix starts and ends at the previous accepting state. It is important to note that, if $\mathcal{A}_\phi$ has at least one accepted run from an initial state, then it has at least one accepted run in the prefix–suffix form [39]. From all such accepted runs of $\mathcal{A}_\phi$, we choose shortest runs in the following way. First, from each initial state, we find the shortest prefixes by finding the shortest paths to all final states. From these final states, we find the shortest suffixes. From all these runs, we select one with the smallest number of states in prefix concatenated with suffix. In all those mentioned earlier, for finding shortest paths, we use Dijkstra's algorithm [44], [45]. Therefore, in our approach, a shortest path corresponds to the fewest number of traversed polytopes. Alternatively, one can consider costs equal to the volumes of the traversed polytopes, or measures of the time spent in each polytope. The details are presented in Algorithm 1.

An empty run resulted from Algorithm 1 has the significance that the formula cannot be satisfied by starting from the current initial state. Let $r_i = r_i(1)r_i(2)r_i(3) \ldots$, $r_i(j) \in Q$ denote the nonempty run of $T$ starting from state $q_i$, i.e., $r_i(1) = q_i$. Since a run $r_i$ is a projected run of $\mathcal{A}_\phi$, any run $r_i$ has a prefix and a suffix with the same significance as stated before. In other words, there exist $n_p^i$ and $n_s^i$ such that for any $j > n_p^i + n_s^i$, $r_i(j) = r_i((j - n_p^i - 1) \bmod n_s^i + n_p^i + 1)$. The number of states in prefix and suffix of $r_i$ are $n_p^i$ and $n_s^i$, respectively, and thus, the run $r_i$ contains at most $n_p^i + n_s^i$ different states.

Under the assumption of an optimal Büchi automaton $\mathcal{B}_\phi$,[1] Algorithm 1 guarantees that in a run $r_i$, none of the states can be succeeded by itself, except for the state of a suffix of length

[1]We say that $\mathcal{B}_\phi$ is optimal if it satisfies the requirement mentioned in the proof of Proposition 2 related to the absence of the "next" operator. In all the tests we performed (by using the implementation from [17]), the obtained $\mathcal{B}_\phi$ was optimal, although the optimality condition we use is not mentioned in [17]. Even if the run $r_i$ was obtained by using a nonoptimal $\mathcal{B}_\phi$, one can collapse all finite successive repetitions of a symbol to a single occurrence of that symbol, and the obtained run will still satisfy the formula $\phi$ (see explanations ending Section II-B).

**Algorithm 1** Find runs of $T$ satisfying $LTL$ formula $\phi$ (inputs: $T$ and $\phi$)

```
Convert φ to an equivalent Büchi automaton B_φ
for all q_k ∈ Q do
  Q_0 = {q_k}
  A_φ = T × B_φ
  r_{A_φ} = ∅
  shortest_length = ∞
  for all s_{A_φ 0} ∈ S_{A_φ 0} do
    for all s_{F_{A_φ}} ∈ F_{A_φ} do
      prefix = shortest_trajectory(s_{A_φ 0} ... s_{F_{A_φ}})
      if prefix ≠ ∅ then
        Remove last state from prefix
        suffix = shortest_trajectory(s_{F_{A_φ}} ... s_{F_{A_φ}})
        if suffix ≠ s_{F_{A_φ}} then
          Remove last state from suffix
        end if
        if suffix ≠ ∅ ∧ length(prefix, suffix) <
        shortest_length then
          r_{A_φ} = prefix, suffix, suffix, ...
          shortest_length = length(prefix, suffix)
        end if
      end if
    end for
  end for
  if r_{A_φ} ≠ ∅ then
    r_k = γ_T(r_{A_φ})
  else
    r_k = ∅ (formula φ cannot be satisfied by any run of T
    starting from q_k)
  end if
end for
```

one (case in which this state will be infinitely repeated). This is in accordance with the allowed semantics for the continuous curves (see Remark 2). Formally, we have:

*Proposition 2:* If $\mathcal{B}_\phi$ is optimal, each nonempty run $r_i = r_i(1)r_i(2)r_i(3)\ldots$ of $T$ satisfies the following property: $r_i(j) \neq r_i(j+1), \forall j \in \mathbb{N} \setminus \{0\}$, $j \neq n_p^i + k\, n_s^i + 1$, $k \in \mathbb{N}$. Moreover, if $n_s^i \geq 2$, $r_i(j) \neq r_i(j+1)$, $\forall j \in \mathbb{N} \setminus \{0\}$.

*Proof:* Let $(r_i(j), s_k)(r_i(j+1), s_l)$ be any two successive states encountered in the run of $\mathcal{A}_\phi$ that was projected to $r_i$, with $j > 0$ and $j \neq n_p^i + k\, n_s^i + 1$, $k \in \mathbb{N}$ (i.e., $r_i(j)$ is not the first state from the suffix of run $r_i$).

Since the run of $\mathcal{A}_\phi$ corresponds to a shortest path, as described before, $(r_i(j), s_k) \neq (r_i(j+1), s_l)$. From Definition 9, this means that $r_i(j) \neq r_i(j+1)$ and/or $s_k \neq s_l$. We want to prove that $r_i(j) \neq r_i(j+1)$. Contradict this hypothesis by considering the case $r_i(j) = r_i(j+1)$, which implies $s_k \neq s_l$. This means that in the Büchi automaton, there is a transition $(s_k, \Pi_{r_i(j)}, s_l) = (s_k, \Pi_{r_i(j+1)}, s_l) \in \rightarrow_{\mathcal{B}_\phi}$. Because $r_i(j) \neq q_0$ (the dummy state $q_0$ was eliminated by projection $\gamma_T$), in the run of $\mathcal{A}_\phi$ there is at least one more state before $(r_i(j), s_k)$. Let us denote this state by $(q_a, s_h)$. Thus, there is a transition $(s_h, \Pi_{r_i(j)}, s_k) \in \rightarrow_{\mathcal{B}_\phi}$. If $\mathcal{B}_\phi$ is optimal, transitions $(s_h, \Pi_{r_i(j)}, s_k)$ and $(s_k, \Pi_{r_i(j)}, s_l)$ imply the existence in $\mathcal{B}_\phi$ of transition $(s_h, \Pi_{r_i(j)}, s_l)$, because LTL$_{-X}$ formula $\phi$ does not contain the "next" operator. Using $r_i(j) = r_i(j+1)$, we obtain $((q_a, s_h), (r_i(j+1), s_l)) \in \rightarrow_{\mathcal{A}_\phi}$. But, this means that the shortest possible trajectory from $(q_a, s_h)$ to $(r_i(j+1), s_l)$ does not contain $(r_i(j), s_k)$, so the hypothesis $r_i(j) = r_i(j+1)$ is false.

If $n_s^i \geq 2$, then there are at least two states in the suffix, so the current run is not ending with an infinite number of repetitions of one state. In this case, the earlier reasoning holds for all states of run $r_i$. ■

To find the runs of $T_g$ satisfying an LTL$_{-X}$ formula $\phi$, we use Algorithm 1 with $T = T_g$. Let $I \subseteq \{1, \ldots, \mathcal{M}\}$ be the set of indexes of all nonempty runs $r_i = r_i(1)r_i(2)r_i(3)\ldots$, $r_i(j) \in Q_g = \mathcal{P}$, $r_i(1) = q_i$.

## VI. Control Strategy

To provide a solution to Problem 1, we restrict the set of initial states of system (3) to

$$x(0) \in \cup_{i \in I} h^{-1}(q_i) \tag{10}$$

where $I \subseteq \{1, \ldots, \mathcal{M}\}$ is the set of indexes of nonempty runs as defined at the end of the previous section.

*Definition 10 (Control strategy):* A control strategy for system (3) corresponding to an LTL$_{-X}$ formula $\phi$ is a tuple $C^\phi = (L, L_0, u, Inv, Rel)$, where:

1) $L = \{l^i_{r_i(j)r_i(j+1)} \mid i \in I,\ j \geq 1\}$ is its set of locations;
2) $L_0 = \{l^i_{q_i r_i(2)},\ i \in I\}$ is the set of initial locations;
3) $Inv$: $L \rightarrow 2^P$, $Inv(l^i_{r_i(j)r_i(j+1)}) = \overline{h^{-1}(r_i(j))}$ gives the invariant for each location;
4) $u$: $L \times P \rightarrow U$ is a map that assigns to each location $l^i_{r_i(j)r_i(j+1)}$ and state $x \in Inv(l^i_{r_i(j)r_i(j+1)})$ a control value $u(l^i_{r_i(j)r_i(j+1)}, x) = u_{r_i(j)r_i(j+1)}(x)$ ($u_{r_i(j)r_i(j+1)}$ are defined in Section IV-B);
5) $Rel \subseteq L \times L$, $Rel = \{(l^i_{r_i(j)r_i(j+1)}, l^i_{r_i(j+1)r_i(j+2)}),\ i \in I,\ j \geq 1,\ r_i(j) \neq r_i(j+1)\}$.

A location $l^i_{r_i(j)r_i(j+1)}$ corresponds to position $j$ in run $r_i$, where $r_i$ is a nonempty run returned by Algorithm 1. According to the structure of runs described in Section V, the set of locations $L$ is finite, even though the runs are infinite. In fact, the number of locations is $\sum_{i \in I}(n_p^i + n_s^i)$, which justifies our search for shortest runs satisfying $\phi$, as described in Section V. A location $l^i_{r_i(j)r_i(j+1)}$ corresponds to driving all states from $\overline{h^{-1}(r_i(j))}$ to $\overline{h^{-1}(r_i(j+1))}$ in finite time (through the common facet of $\overline{h^{-1}(r_i(j))}$ and $\overline{h^{-1}(r_i(j+1))}$) if $r_i(j) \neq r_i(j+1)$, or to keeping the state of the system in $h^{-1}(r_i(j))$ for all times if $r_i(j) = r_i(j+1)$, by using the control $u_{r_i(j)r_i(j+1)}(x)$. Note that there can be several locations mapped to the same physical region $\overline{h^{-1}(q)}$, $q \in Q$. These can correspond to different runs of $T_g$ passing through $q$ or to locations of the same run passing through $q$ at different times and with different successors.

The semantics of control strategy from Definition 10 applied to system (3) with initial states (10) are as follows: starting from $x(0) \in h^{-1}(q_i)$ and location $l = l^i_{q_i r_i(2)} \in L_0$, feedback controller $u(l, x)$ is applied to system (3) as long as the state $x \in Inv(l)$. When (and if) $x \notin Inv(l)$, then the location of

$C^\phi$ is updated to $l'$ according to $(l, l') \in Rel$, and the process continues.

*Remark 3:* From the given semantics of the control strategy, it follows that the control is well defined on common facets: the one from the polytope that is left is always used. Also, with controllers $u_{q_i q_i}$ and $u_{q_i q_j}$ designed according to Corollaries 1 and 2, the produced trajectories are consistent with Definition 4 in the sense of Remark 1.

We are now ready to provide a solution to Problem 1:

*Theorem 3:* All trajectories of system (3), with feedback control strategy given by Definition 10 and set of initial states as in (10), satisfy the LTL$_{-X}$ formula $\phi$ and stay inside $P$ for all times.

*Proof:* The proof follows from the construction of $C^\phi$ from Definition 10, the satisfaction of an LTL$_{-X}$ formula by a continuous trajectory given in Definition 5 and Corollaries 1 and 2 of Theorems 1 and 2.

Let $a$ denote a trajectory of system (3) with control strategy $C^\phi$ starting from an arbitrary initial state $a(0) \in h^{-1}(q_i),\ i \in I$. Then, by Definition 4, the corresponding word $w_a$ has $w_a(1) = \Pi_{q_i}$. The applied control $u(l^i_{q_i r_i(2)}, a(0))$ is designed in accordance with Corollary 1 if $q_i \neq r_i(2)$, or Corollary 2 if $q_i = r_i(2)$.

In the first case, trajectory $a(t)$ is guaranteed to leave $Inv(l^i_{q_i r_i(2)}) = \overline{h^{-1}(q_i)}$ through the interior of facet $\overline{h^{-1}(q_i)} \cap \overline{h^{-1}(r_i(2))}$ and hit $h^{-1}(r_i(2))$ in finite time. From Definition 4, it follows that the symbol $w_a(2) = \Pi_{r_i(2)}$ is added to $w_a$. Note that $w_a(1)w_a(2)$ are also the first two symbols of word generated by run $r_i$ of $T_g$. The same reasoning holds for any location $l^i_{r_i(j) r_i(j+1)} \in L$ with $r_i(j) \neq r_i(j+1)$. In the second case ($q_i = r_i(2)$), by Corollary 2, trajectory $a$ will always remain inside $h^{-1}(q_i)$ by evolving under control $u(l^i_{q_i q_i}, x)$. The word $w_a$ will be $w_a = \Pi_{q_i}\Pi_{q_i} \ldots$. From Definition 10, location $l^i_{q_i q_i}$ does not have any outgoing relation in $Rel$. From Proposition 2, we have that $r_i = q_i q_i \ldots$, so the corresponding word of $T_g$ is $w_a$. A similar reasoning can be used for any location $l^i_{r_i(j) r_i(j)} \in L$.

We conclude that any trajectory $a$ of (3) with initial state in (10) and control strategy given by Definition 10 will generate a word $w_a$, which is also a word of $T_g$ satisfying formula $\phi$. Any trajectory also satisfies $a(t) \subset \cup_{i \in I} \cup_{j \in \mathbb{N} \setminus \{0\}} \overline{h^{-1}(r_i(j))} \subseteq P$, $\forall t \geq 0$, and the theorem is proved. ∎

*Remark 4:* It is possible that the solution trajectories visit some states more than once, and have different velocities at the same state at different times. Therefore, the obtained feedback control strategy is a dynamic extension. The feedback controllers will be piecewise affine and with a thin set of discontinuities—the common facets of full-dimensional subpolytopes of $P$. The generated trajectories will be piecewise smooth and everywhere continuous.

To implement the control strategy described in Definition 10, we have, in general, infinitely many choices of controllers of the types $u_{q_i q_i}$ and $u_{q_i q_j}$. Indeed, for any polytope, Corollaries 1 and 2 return whole polyhedral sets of allowed controls at vertices. In order to construct a controller according to (7) and (8), we need to choose a control at each vertex. To this goal, we solve a set of (maximization) linear programs obtained by attaching a cost to each vertex. If a controller of type $u_{q_i q_j}$ is desired in $h^{-1}(q_i)$, then the costs corresponding to the vertices of $h^{-1}(q_i)$ are the projections of the controls at the vertices along the unit vector connecting the center of $h^{-1}(q_i)$ to the center of $h^{-1}(q_j)$. If a controller of type $u_{q_i q_i}$ is desired in $h^{-1}(q_i)$, then the cost at a vertex is the projection of the control at the vertex along the unit vector from the vertex to the center of $h^{-1}(q_i)$.

## VII. Discussion

Our approach to solving Problem 1 is obviously conservative. If the planning algorithm does not find any solution, this does not mean that there does not exist initial states and feedback controllers producing trajectories satisfying the formula. In other words, our solution to Problem 1 is *not complete*. There are three sources of conservativeness in our approach. First, we look for whole sets (full-dimensional polytopes) of initial states instead of investigating isolated ones. Second, we restrict our attention to affine feedback controllers, as opposed to allowing for any type of controllers. Third, Theorem 1 and Corollary 1 provide sufficient conditions for the existence of controllers, as opposed to equivalent conditions.

On the positive side, working with sets of states instead of isolated states provides robustness with respect to uncertainty in initial conditions and measurement of the current state. As proved in [16], Theorem 1 can be replaced with a very similar result providing equivalent conditions for the existence of affine controllers if full-dimensional simplices are considered instead of full-dimensional polytopes. Therefore, if $P$ was triangulized instead of partitioned into arbitrary polytopes, the third source of conservativeness would be eliminated. Another advantage of using simplices instead of polytopes would be the fact that we could produce smooth trajectories everywhere by matching the choice of controls at vertices on adjacent simplices [30]. We chose polytopes as opposed to simplices for two reasons. First, as far as we know, there does not exist algorithms for triangulation in dimension larger than 2 that preserve linear constraints (we need to produce proposition preserving partitions when we construct $T_g$). Second, triangulations can produce an explosion in the number of states of $T_g$.

On the complexity of our approach, we can easily construct an upper-limit for the number of states in the product automaton $\mathcal{A}_\phi$. If $n$ is the number of predicates from (4), then transition system $T_g$ has maximum $2^n$ states (if $n > N$, there will be less than $2^n$ states), and the Büchi automaton $\mathcal{B}_\phi$ obtained using LTL2BA has maximum $n \times 2^n$ states [17]. Thus, the product automaton (including the dummy state $q_0$) has at most $n \times 2^n \times (2^n + 1)$ states. Note that this limit does not depend on the dimension $N$ of the state space. However, if $n > N$, which is usually the case, then the size of the state space plays a role in the number of states of $T_g$. Indeed, as $N$ increases, a constant number of half spaces $n$ can define an increasing number of feasible subpolytopes. The good news is that, in practice, the upper limits for both the Büchi automaton and the generator transition system are almost never reached.

Finally, it is worth mentioning that, in our implementation, we use an iterative procedure to construct the set of feasible

subpolytopes, while at the same time taking into consideration new constraints. This way, since usually $\mathcal{M} \ll 2^n$ (especially for large values for $n$), we end up with testing a number of proposition combinations much smaller than $2^n$.

## VIII. Implementation Notes and Simulation Results

We implemented our approach as a user friendly software package for linear temporal logic control of linear systems LTLCon under Matlab. The tool, which is freely downloadable from [18], takes as input the polytope $P$, the matrices $A, B$, and $b$ of system (3), and the LTL$_{-X}$ formula $\phi$. If it finds a solution, it plots the produced trajectories corresponding to user defined initial states. Even though transparent to the user, LTLCon also uses two free packages. The first one is a mex-file calling CDD in Matlab [46], and it is used to convert a polytope expressed in form (1) to form (2) and vice versa. The second one is LTL2BA [17], which is used to convert an LTL formula to a Büchi automaton.

To illustrate the use of LTLCon, we first consider a 2-D case ($N = 2$), chosen for simplicity of graphical representation. We considered the following numerical values for system (3)

$$\dot{x} = \begin{bmatrix} 0.2 & -0.3 \\ 0.5 & -0.5 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad x \in P,\ u \in U. \tag{11}$$

Polytope $P$ is specified in form (2), as the intersection of eight closed half spaces, defined by: $a_1 = [-1\ \ 0]^T$, $b_1 = -5$, $a_2 = [1\ \ 0]^T$, $b_2 = -7$, $a_3 = [0\ \ -1]^T$, $b_3 = -3$, $a_4 = [0\ \ 1]^T$, $b_4 = -6$, $a_5 = [-3\ \ -5]^T$, $b_5 = -15$, $a_6 = [1\ \ -1]^T$, $b_6 = -7$, $a_7 = [-1\ \ 2.5]^T$, $b_7 = -15$, $a_8 = [-2\ \ 2.5]^T$, $b_8 = -17.5$. Control constraints are captured by the set $U = [-2, 2] \times [-2, 2]$.

We define a set $\Pi$ containing ten predicates, as in (4) and (5), where: $c_1 = [0\ \ 1]^T$, $d_1 = 0$, $c_2 = [1\ \ -1]^T$, $d_2 = 0$, $c_3 = [4\ \ 1]^T$, $d_3 = 12$, $c_4 = [4\ \ -7]^T$, $d_4 = 34$, $c_5 = [-2\ \ -1]^T$, $d_5 = 4$, $c_6 = [-1\ \ -12]^T$, $d_6 = 31$, $c_7 = [-1\ \ -1]^T$, $d_7 = 11$, $c_8 = [1\ \ 0]^T$, $d_8 = -3$, $c_9 = [0\ \ -1]^T$, $d_9 = -1.5$, $c_{10} = [-6\ \ -4.5]^T$, $d_{10} = -12$.

There are 33 feasible full-dimensional subpolytopes in $P$, and therefore, 33 states in $T_g$. Fig. 2 depicts the bounding polytope $P$, the vector field given by the drift of system (11), the predicates $\pi_i$, $i = 1, \ldots, 10$, and the feasible subpolytopes corresponding to states $q_i$, $i = 1, \ldots, 33$ of $T_g$. The lines connecting the centroids of the polytopes in Fig. 2 represent the transitions of $T_g$, with the following convention: for all $i \neq j$: a full line means that $(q_i, q_j), (q_j, q_i) \in \rightarrow_g$; a dashed line means that $(q_i, q_j) \in \rightarrow_g$ for $i < j$; a dotted line means that $(q_i, q_j) \in \rightarrow_g$ for $i > j$. A self-transition $(q_i, q_i) \in \rightarrow$ is represented by a star in the center of $h^{-1}(q_i)$.

We have chosen an LTL$_{-X}$ formula inspired from robot motion planning, which involves visiting a sequence of three regions infinitely often, while always avoiding three obstacles. The regions to be visited are, in order: $r_1 = h^{-1}(q_1)$, $r_2 = \bigcup_{i \in \{20,21,29\}} h^{-1}(q_i)$, and $r_3 = h^{-1}(q_{32})$. The obstacles are represented by the polyhedral regions $o_1 = \bigcup_{i \in \{13,14,16,17,18\}} h^{-1}(q_i)$, $o_2 = \bigcup_{i \in \{19,28\}} h^{-1}(q_i)$, and $o_3 = h^{-1}(q_{10})$. All regions to be visited and obstacles are represented in Fig. 2. The LTL$_{-X}$ formula can be written as $\phi = \square(\diamondsuit(r_1 \wedge \diamondsuit(r_2 \wedge \diamondsuit r_3)) \wedge \neg(o_1 \vee o_2 \vee o_3))$.

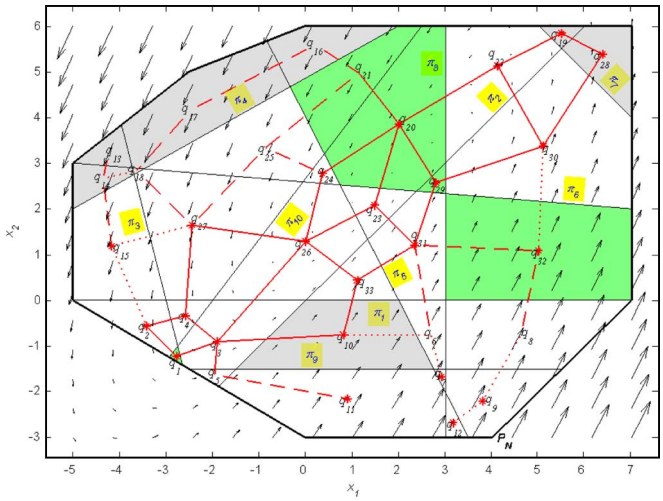



Fig. 2. The arrows represent the drift vector field of system (11). The white boxes mark the half-spaces corresponding to atomic propositions $\pi_i$, $i = 1, \ldots, 10$. The regions to be visited are light gray, while the obstacles are gray.

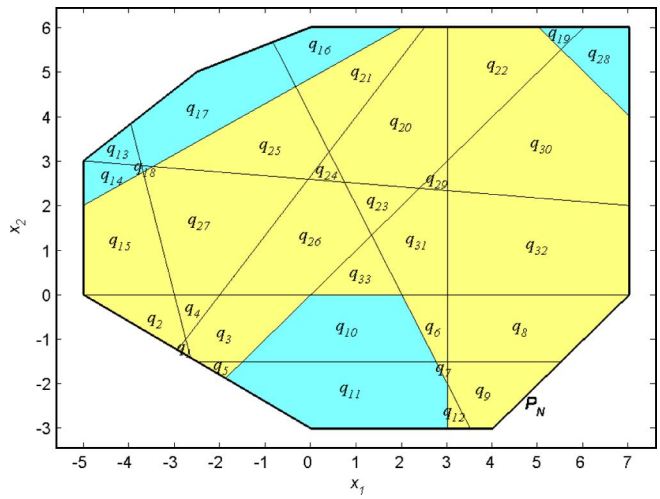
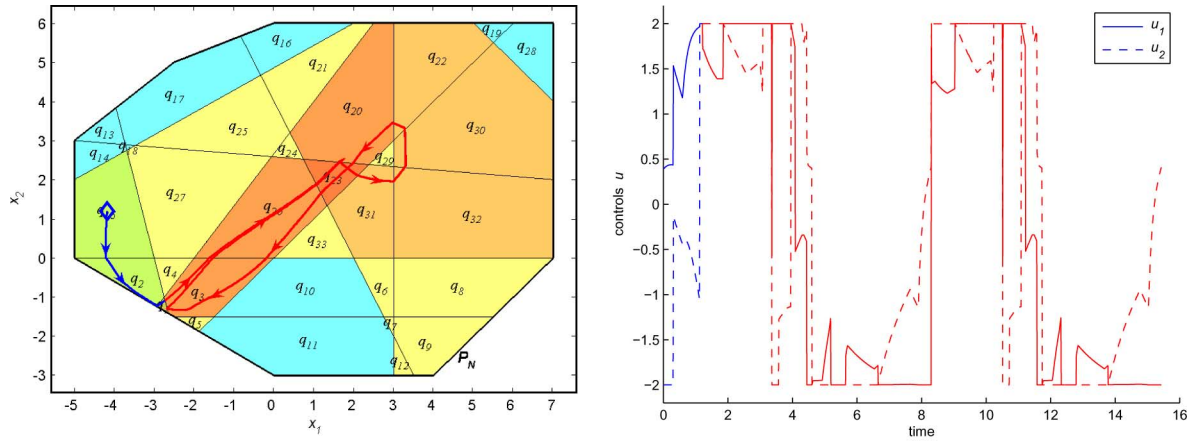



Fig. 3. Union of the white polytopes represents the set of initial states from which there exist continuous trajectories satisfying formula $\phi$.

By expressing interesting regions $r_i$ and $o_i$, $i = 1, 2, 3$ in terms of predicates $\pi_j$, $j = 1, \ldots, 10$, we obtain $\phi = \square(\diamondsuit((\pi_3 \wedge \pi_{10}) \wedge \diamondsuit((\neg\pi_4 \wedge \pi_5 \wedge \pi_6 \wedge \pi_8) \wedge \diamondsuit(\neg\pi_1 \wedge \neg\pi_6 \wedge \neg\pi_8))) \wedge \neg(\pi_4 \vee \pi_7 \vee (\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_5 \wedge \pi_9)))$.

The set of initial states from which there exist continuous trajectories satisfying the formula is the union of the white polytopes in Fig. 3. The set of initial states of $T_g$ from which there exist runs satisfying the formula are the corresponding labels. Run $r_{15}$ of $T_g$ starting from $q_{15}$ and satisfying $\phi$ is presented in Fig. 4(a). The prefix of run $r_{15}$ of $T_g$ is $q_{15}q_2$ (shown as light gray polytopes), while the suffix is $q_1q_3q_{26}q_{23}q_{20}q_{23}q_{31}q_{32}q_{30}q_{22}q_{20}q_{23}q_{26}q_3$ (gray polytopes). A continuous trajectory starting from the centroid of polytope $h^{-1}(q_{15})$, $x_0 = [-4.17\ \ 1.19]^T$, is also shown in Fig. 4(a). The




(a) (b)

Fig. 4. Simulation results. (a) Run $r_{15}$ returned by Algorithm 1: light gray polytopes correspond to prefix, while gray polytopes correspond to suffix. The continuous trajectory starting from $x_0 = [-4.17 \;\; 1.19]^T$ (marked with a diamond) is colored in black for its prefix part and in gray for its suffix part. (b) Applied controls: $u_1$ (full line) and $u_2$ (dashed line)—the black parts correspond to prefix, while the gray parts correspond to suffix.

part corresponding to the prefix is black, while the suffix is shown in gray.

The previous case study was run on a Pentium 4 (2.66 GHz) machine with 1 GB RAM, Windows XP, and Matlab 7. The transition system $T_g$ with 33 states was created in about 0.9 s. The Büchi automaton had nine states and was created in 2.2 s. The desired runs of $T_g$ were obtained in about 11 s.

We also ran a 4-D example ($N = 4$), with $P$ defined by nine hyperplanes and $\Pi$ containing $n = 15$ predicates. There were $\mathcal{M} = 295$ states in $T_g$—its construction took 68 s. A tesselation using the intersection points between hyperplanes defining the predicates would yield 17509 tetrahedra. As explained before, these simplices are not suitable for our problem; but, even if they were, a transition system with so many states would be inefficient from a computational point of view.

## IX. Conclusion

In this paper, we described a fully automated framework for control of linear systems from specifications given in terms of LTL$_{-X}$ formulas over linear predicates in its state variables. We expect that the method will find applications in several areas of engineering, where linear systems are used for modeling and temporal logic for specifying correctness and performance. Such areas include robotic motion planning and control of gene networks. Future directions of research include the extension of these techniques to continuous multiaffine systems [47], piecewise affine, or multiaffine hybrid systems, and the inclusion of branching time logical specifications.

## References


[1] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science: Formal Models and Semantics,* vol. B, J. van Leeuwen, Ed. Amsterdam, The Netherlands: North Holland/MIT Press, 1990, pp. 995–1072.

[2] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA: MIT Press, 2000.

[3] B. Shults and B. Kuipers, "Proving properties of continuous systems: Qualitative simulation and temporal logic," *Artif. Intell.*, vol. 92, no. 1–2, pp. 91–130, 1997.

[4] G. Brajnik and D. Clancy, "Focusing qualitative simulation using temporal logic: Theoretical foundations," *Ann. Math. Artif. Intell.*, vol. 22, no. 1–2, pp. 59–86, 1998.

[5] P. Tabuada and G. Pappas, "Model checking LTL over controllable linear systems is decidable," in *Hybrid Systems: Computation and Control,* vol. 2623 (Ser. Lecture Notes in Computer Science), O. Maler and A. Pnueli, Eds. New York: Springer-Verlag, 2003, pp. 498–513.

[6] T. Moor and J. Davoren, "Robust controller synthesis for hybrid systems using modal logic," in *Hybrid Systems: Computation and Control,* vol. 2034 (Ser. Lecture Notes in Computer Science), C. J. Tomlin and M. R. Greenstreet, Eds. New York: Springer-Verlag, 2001, pp. 433–446.

[7] M. Antoniotti and B. Mishra, "Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 21–27 May, 1995, vol. 2, pp. 1441–1446.

[8] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *Proc. 43rd IEEE Conf. Dec. Control*, 14–17 Dec. 2004, vol. 1, pp. 153–158.

[9] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot motion planning: A timed automata approach," in *Proc. 2004 IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, pp. 4417–4422.

[10] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proc. 2005 IEEE Conf. Dec. Control*, Seville, Spain, pp. 4885–4890.

[11] M. Kloetzer and C. Belta, "LTL planning for groups of robots," in *Proc. IEEE Int. Conf. Netw., Sensing, Control*, Ft. Lauderdale, FL, 23–25 Apr., 2006, pp. 578–583.

[12] M. Kloetzer and C. Belta, "Hierarchical abstractions for robotic swarms," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 15–19 May 2006, pp. 952–957.

[13] 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.*, L. H. R. B. Altman, A. K. Dunker, and T. Klein, Eds. Lihue, Hawaii, 2003, pp. 116–127.

[14] 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 *E. coli*," presented at the 13th Int. Conf. Intell. Syst. Molecular Biol., Detroit, MI, 2005.

[15] X. Koutsoukous, P. Antsaklis, J. Stiver, and M. Lemmon, "Supervisory control of hybrid systems," in *Proc. IEEE, Special Issue Hybrid Syst.*, Jul. 2000, vol. 88, no. 7, pp. 1026–1049.

[16] L. Habets and J. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, pp. 21–35, 2004.

[17] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proc. 13th Conf. Comput. Aided Verification (CAV'01)* (Ser. Lecture Notes in Computer Science), H. C. G. Berry and A. Finkel, Eds. New York: Springer-Verlag, 2001, vol. 2102, pp. 53–65.

[18] M. Kloetzer and C. Belta, "LTLCon, a Matlab package for control of linear systems from linear temporal logic specifications," 2005 [Online]. Available: http//iasi.bu.edu/∼software/LTL-control.htm
[19] J. Davoren, V. Coulthard, N. Markey, and T. Moor, "Non-deterministic temporal logics for general flow systems," in *Proc. 7th Int. Workshop Hybrid Syst.: Comput. Control*, Philadelphia, PA, 2004, pp. 280–295.
[20] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.
[21] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "Hybrid automata: An algorithmic approach to specification and verification of hybrid systems," *Theoretical Comput. Sci.*, vol. 138, pp. 3–34, 1995.
[22] A. Puri and P. Varaiya, "Decidability of hybrid systems with rectangular inclusions," *Comput. Aided Verification*, vol. 818, pp. 95–104, 1994.
[23] G. Lafferriere, G. J. Pappas, and S. Sastry, "O-minimal hybrid systems," *Math. Control, Signals Syst.*, vol. 13, no. 1, pp. 1–21, Mar. 2000.
[24] 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.
[25] G. J. Pappas, "Bisimilar linear systems," *Automatica*, vol. 39, no. 12, pp. 2035–2047, 2003.
[26] M. Broucke, "A geometric approach to bisimulation and verification of hybrid systems," in *Hybrid Systems: Computation and Control,* (Ser. Lectures Notes in Computers Science), F. W. Vaandrager and J. H. van Schuppen, Eds. New York: Springer-Verlag, 1999, vol. 1569, pp. 61–75.
[27] E. Haghverdi, P. Tabuada, and G. Pappas, "Bisimulation relations for dynamical and control systems," *Electron. Notes Theoretical Comput. Sci.*, vol. 69, pp. 1–17, 2003.
[28] A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," presented at the 5th Int. Workshop Hybrid Syst.: Comput. Control, Stanford, CA, 2002.
[29] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *Proc. Hybrid Syst.: Comput. Control: 9th Int. Workshop,* vol. 3927 (Ser. Lecture Notes in Computer Science), J. Hespanha and A. Tiwari, Eds. Berlin/Heidelberg, Germany: Springer-Verlag, 2006, pp. 348–362.
[30] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot planning and control in polygonal environments," *IEEE Trans. Robot.*, vol. 21, no. 5, pp. 864–874, Oct. 2005.
[31] T. Motzkin, H. Raiffa, G. Thompson, and R. M. Thrall, "The double description method," in *Contributions Theory Games,* vol. 2, H. Kuhn and A. Tucker, Eds. Princeton, NJ: Princeton University Press, 1953.
[32] K. Fukuda, "CDD/CDD+ package," [Online]. Available: http://www.cs.mcgill.ca/∼fukuda/soft/cdd_home/cdd.html
[33] C. W. Lee, "Subdivisions and triangulations of polytopes," in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O. Rourke, Eds. Boca Raton, NY: CRC Press, 1997, pp. 271–290.
[34] G. M. Ziegler, "Lectures on polytopes," in *Graduate Texts in Mathematics*. Berlin, Germany: Springer-Verlag, 1995, vol. 152.
[35] S. Fortune, "Voronoi diagrams and Delaunay triangulations," in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O. Rourke, Eds. Boca Raton, NY: CRC Press, 1997, pp. 377–388.
[36] B. Jeannet, "Convex polyhedra library," Verimag, Grenoble, France, Tech. Rep. 1016, 1999.
[37] D. K. Wilde, "A library for doing polyhedral operations," IRISA, Rennes, France, Tech. Rep. Publication Interne 785, 1993.
[38] C. Belta and L. Habets, "Constructing decidable hybrid systems with velocity bounds," in *Proc. 43rd IEEE Conf. Dec. Control*, Paradise Island, Bahamas, 14–17 Dec. 2004, vol. 1, pp. 467–472.
[39] G. Holzmann, *The SPIN Model Checker, Primer and Reference Manual*. Reading, MA: Addison-Wesley, 2004.
[40] P. Wolper, M. Vardi, and A. Sistla, "Reasoning about infinite computation paths," in *Proc. 24th IEEE Symp. Found. Comput. Sci.*, E. N. *et al.*, Ed. Tucson, AZ, 1983, pp. 185–194.
[41] R. Gerth, D. Peled, M. Vardi, and P. Wolper, "Simple on-the-fly automatic verification of linear temporal logic," in *Proc. 15th IFIP WG6.1 Int. Symp. Protocol Spec., Testing Verification XV*. London, U.K.: Chapman & Hall, 1996, pp. 3–18.
[42] P. Wolper, "Constructing automata from temporal logic formulas: A tutorial," in *Lectures Formal Methods Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science,* vol. 2090 (Ser. Lecture Notes in Computer Science), H. H. E. Brinksma and J. Katoen, Eds. New York: Springer-Verlag, 2001, pp. 261–277.
[43] G. D. Giacomo and M. Vardi, "Automata-theoretic approach to planning for temporally extended goals," in *Proc. 5th Eur. Conf. Planning (ECP '99)*, vol. 1809, pp. 226–238.
[44] E. Dijkstra, "A note on two problems in connexion with graphs," in *Numerische Mathematik*. Amsterdam, The Netherlands: Mathematisch Centrum, 1959, vol. 1, pp. 269–271.
[45] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA and New York: MIT Press and McGraw-Hill Book Company, 2001.
[46] F. Torrisi and M. Baotic, "Matlab interface for the CDD solver," [Online]. Available: http://control.ee.ethz.ch/∼hybrid/cdd.php
[47] C. Belta and L. Habets, "Control of a class of nonlinear systems on rectangles," *IEEE Trans. Automat. Control*, vol. 51, no. 11, pp. 1749–1759, Nov. 2006.

**Marius Kloetzer** (S'05) received the B.S. and M.Sc. degrees in computer science from the Technical University of Iasi, Iasi, Romania, in 2002 and 2003, respectively. He is currently working toward the Ph.D. degree in systems engineering at the Center for Information and Systems Engineering, Boston University, Boston, MA.

His current research interests include robot motion planning using discrete abstractions, linear temporal logic, and hybrid systems.

**Calin Belta** (S'00–M'03) received the B.S. and M.Sc. degrees in control and computer science from the Technical University of Iasi, Iasi, Romania, in 1995 and 1996, respectively, the second M.Sc. degree in electrical engineering from Louisiana State University, Baton Rouge, in 1999, and the third M.Sc. and Ph.D. degrees in mechanical engineering from the University of Pennsylvania, Philadelphia, in 2002 and 2003, respectively.

He is currently an Assistant Professor at the Center for Information and Systems Engineering, Boston University, Boston, MA. His current research interests include planning and control for formations of robots, hybrid systems, and bio-molecular networks.

Dr. Belta is the recipient of the National Science Foundation CAREER Award in 2005, the Fulbright Study Award in 1997, and was the Valedictorian of his class in 1995. He received the Best Paper Award at the International Conference on Systems Biology in 2004 and was a Finalist for the American Society of Mechanical Engineers Design Engineering Technical Conference Best Paper Award in 2002 and for the Anton Philips Best Student Paper Award at the IEEE International Conference on Robotics and Automation in 2001.