A Fully Automated Framework for Control of Linear Systems from LTL Specifications*

Marius Kloetzer and Calin Belta

Center for Information and Systems Engineering,
Boston University, 15 Saint Mary's Street,
Boston, MA 02446
{kmarius, cbelta}@bu.edu

Abstract. We consider the following problem: given a linear system and an LTL_{-X} 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 model checking, we determine runs of the transition system satisfying the formula. Third, we generate the control strategy. Illustrative examples are included.

1 Introduction

Temporal logic [1] is the natural framework for specifying and verifying the correctness of computer programs. However, due to their resemblance to natural language, their 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 [2, 3, 4]. In the control-theoretic community, a framework for specifying and controlling the behavior of a discrete linear system has been developed in [5]. The use of temporal logic for task specification and controller synthesis in mobile robotics has been advocated as far back as [6], and recent results include [7, 8, 9]. 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 [10, 11].

We consider the following problem: *given a linear system $\dot{x} = Ax + b + Bu$ with polyhedral control constraints U, and given an arbitrary LTL_{-X} formula ϕ over an arbitrary set of linear predicates, find initial states and a feedback control strategy u so that the corresponding trajectories of the closed loop system verify the formula ϕ, while staying inside a given full-dimensional polytope P_N.*

Our approach to solving the above problem can be summarized as the following three steps. In the first step, we construct a finite state "generator" transition system T_g.

* This work is partially supported by NSF CAREER 0447721 and NSF 0410514.

J. Hespanha and A. Tiwari (Eds.): HSCC 2006, LNCS 3927, pp. 333–347, 2006.

Its states are the equivalence classes produced by the feasible full-dimensional subpolytopes of P_N determined by the linear predicates appearing in the formula ϕ. 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 [12]. In the second step, we produce runs of T_g that satisfy formula ϕ. This is in essence a model checking problem, and we use standard tools based on Büchi automata [13]. In the third step, we construct a feedback "control strategy", which leads to a closed loop hybrid system, whose continuous trajectories satisfy formula ϕ. We implemented our approach as a user friendly software package LTLCON [14] under Matlab.

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 [4] 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 [15], followed by multi-rate automata [16], and rectangular hybrid automata [17]. Other classes of systems for which finite bisimulation quotients exist are identified in [18]. The interested reader is referred to [19] for an excellent review of all these works. Linear dynamics are studied in [20], while nonlinear systems are considered in [21, 22]. Quotients that only simulate a continuous or hybrid system and can be used for conservative analysis are developed in [23].

This paper is inspired from [5, 9]. The problem of controller synthesis from LTL specifications for discrete-time continuous-space linear systems with semi-linear 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 existence and computability. Specifically, it is shown that the iterative (partitioning) bisimulation algorithm [18] terminates and each step is computable. However, no computational formulas for the controllers are provided. Another contribution of [5] is setting up the framework for producing runs of the finite quotient satisfying an LTL formula. This framework is further refined in [9], where the authors study the problem of controlling a planar robot in a polygon so that its trajectory satisfies an LTL_{-X} formula. In [9], 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 construction of vector fields, the authors use the algorithms developed in [24].

This paper extends the results of [5, 9] 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 [12], and an invariance theorem stated 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 [9] 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 Preliminaries

2.1 Polytopes

Let $N \in \mathbb{N}$ and consider the N - dimensional Euclidean space $\mathbb{R}^N$. A full dimensional *polytope* P_N 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_N 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_N. A polytope is completely described by its set of vertices:

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

where $conv$ denotes the convex hull. Alternatively, P_N 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

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

Forms (1) and (2) are referred to as V- and H- representations of the polytope, respectively. Given a full dimensional polytope P_N, there exist algorithms for translation from representation (1) to representation (2) [25, 26]. A *face* of P_N is the intersection of P_N 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. A $(N-1)$-face obtained by intersecting P_N with one of its supporting hyperplanes is called a *facet*. The vertices of P_N are 0-faces. We denote by $int(P_N)$ the set of points of P_N 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_N, $int(F)$ is defined analogously, with the observation that F is 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* [27]. In other words, for any full dimensional polytope P_N, there exist full dimensional simplices $S_1, \ldots, S_L$ such that: (i) $P_N = \bigcup_{i=1}^{L} S_i$, (ii) $S_i \bigcap S_j$ is either empty or a common face of S_i and S_j, for all $i, j = 1, \ldots, L$, $i \neq j$, and (iii) The set of vertices of simplex S_i is a subset of $\{v_1, \ldots, v_M\}$, for all $i = 1, \ldots, L$.

2.2 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, Π is a finite set of atomic propositions, and $\models \subseteq Q \times \Pi$ is a satisfaction relation.*

In this work, we assume that the transition system is *finite* (Q is finite). For an arbitrary proposition $\pi \in \Pi$, we define $[[\pi]] = \{q \in Q | q \vDash \pi\}$ as the set of all states satisfying it. Conversely, for an arbitrary state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \mid q \vDash \pi\}$, $\Pi_q \in 2^{\Pi}$, 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 Π is recursively defined as follows:*

- *Every atomic proposition π_i, $i = 1, \ldots, K$ is a formula, and*
- *If ϕ_1 and ϕ_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 ϕ at position $i \in \mathbb{N}$ of word w, denoted by $w(i) \vDash \phi$, is defined recursively as follows:*

- *$w(i) \vDash \pi$ if $\pi \in w(i)$,*
- *$w(i) \vDash \neg\phi$ if $w(i) \nvDash \phi$,*
- *$w(i) \vDash \phi_1 \vee \phi_2$ if $w(i) \vDash \phi_1$ or $w(i) \vDash \phi_2$,*
- *$w(i) \vDash \phi_1 \mathcal{U} \phi_2$ if there exist a $j \geq i$ such that $w(j) \vDash \phi_2$ and for all $i \leq k < j$ we have $w(k) \vDash \phi_1$*

A word w satisfies an LTL_{-X} formula ϕ, written as $w \vDash \phi$, if $w(1) \vDash \phi$.

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) ϕ_2 will eventually become true and ϕ_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 $\Box\phi = \phi\mathcal{U}\bot$, respectively. Formula $\Diamond\phi$ means that ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true at all positions of w. More expressiveness can be achieved by combining the temporal operators. Examples include $\Box\Diamond\phi$ (ϕ is true infinitely often) and $\Diamond\Box\phi$ (ϕ becomes eventually true and stays true forever).

LTL [1], the most used propositional linear temporal logic, is richer than LTL_{-X} in the sense that it allows for an additional temporal operator $\bigcirc$, which is called the 'next' operator. Formally, the syntax of LTL is obtained by adding "$\bigcirc\phi_1$" to Definition 2 and its semantics is defined by adding "$w(i) \vDash \bigcirc\phi$ if $w(i+1) \vDash \phi$" 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. 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 3, a word corresponding to a continuous

trajectory will never have a finite number of successive repetitions of a symbol. In Section 4.3, we produce runs which will either have one or infinitely many successive appearances of a symbol (or finite sequences of symbols).

3 Problem Formulation and Approach

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

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

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 Π be a set of atomic propositions given as arbitrary strict linear inequalities in $\mathbb{R}^N$. Formally:

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

where each proposition π_i, $i = 1, \ldots, n$, denotes an open half-space of $mathbbR^N$:

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

The polytope P_N 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, P_2 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 Π 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 ϕ over Π, 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 ϕ, while always staying inside P_N.

To fully specify Problem 1, we need to define the satisfaction of an LTL_{-X} formula ϕ on Π by a trajectory of (3), which can be seen as a continuous curve $\alpha : [0, \infty) \to \mathbb{R}^N$. This curve can, in general, be non-smooth and can have self-intersections. For each symbol $\Theta \in 2^\Pi$, 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 α is the sequence $w_\alpha = w_\alpha(1)$ $w_\alpha(2)w_\alpha(3)\ldots$, $w_\alpha(k) \in 2^\Pi$, $k \geq 1$, generated such that the following rules are satisfied for any $\tau \geq 0$ and any $k \in \mathbb{N}^*$:*

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

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. Note that Definition 4 captures the situations when a trajectory hits a sink region, leave it and eventually come back and remains there, as well as Zeno-type behaviors, when a trajectory visits two adjacent regions infinitely often.

Remark 1. On the well posedness of Definition 4, first note that our assumption that trajectories of system (3) always stay inside P_N 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, the predicates in (4) are given by strict linear inequalities, Definition 4 makes sense only if $c_i^T \alpha(0) + d_i \neq 0$ and $c_i^T \bar{\alpha} + d_i \neq 0$, where $\bar{\alpha} = \lim_{t \to \infty} \alpha(t)$ (if it exists), for all $i = 1, \ldots, n$. Third, Definition 4 is a proper characterization of satisfaction of sets of predicates from Π by $\alpha(t)$ as time evolves only if there does not exist $t_1 < t_2$ and $i = 1, \ldots, n$ such that $c_i^T \alpha(t) + d_i = 0$, 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 4.1 and 5.

Remark 2. According to Definition 4, the word w_α produced by a trajectory $\alpha(t)$ does not contain a finite number of successive repetitions of a symbol, which suggests using LTL without the 'next' operator, as stated in Section 2.2.

Definition 5. *A trajectory* $\alpha : [0, \infty) \to \mathbb{R}^N$ *of (3) satisfies* LTL_{-X} *formula* ϕ*, written as* $\alpha \vDash \phi$*, if and only if* $w_\alpha \vDash \phi$*, where* w_α *is the word generated by* α *in accordance with Definition 4.*

4 The Generator Transition System

4.1 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 [12]]. 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 [12]]. 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) $n_1^T(Av_j + Bu_j + a) > 0$.

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 (i)* $\forall t \in [0, T_0]: \; x(t) \in P$*, (ii)* $x(T_0) \in F_1$*, and (iii)* $n_1^T \dot{x}(T_0) > 0$.

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 (iii) 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. See [28].

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,...,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:

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

where the control in each simplex is given by [29]:

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

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 [12, 29]. 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, formula (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 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 is easy to see 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 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), (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), (8).*

4.2 Construction of the Generator Transition System

Assume the polytope P_N is given in the inequality form (2). Assume there are $\mathcal{M}$, $1 \leq \mathcal{M} \leq 2^n$ 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_N and corresponds to a feasible combination of all predicates from Π inside P_N). 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 $P_{N\_} = int(P_N) \backslash \bigcup_{i=1}^{n} \{x \in \mathbb{R}^n | c_i^T x + d_i = 0\}$. 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_N. 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 $\bigcup_{i=1}^{\mathcal{M}} \overline{h^{-1}(q_i)} = P_N$.

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

- $Q_g = Q_{g0} = \mathcal{P}$,
- *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,*
- *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,*
- *$\Pi_g = \Pi$, with Π as defined in (4),*
- *$q \vDash_g \pi_i \in \Pi$ 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 non-emptiness of polyhedral sets (since U is polyhedral), for which there exists several powerful algorithms.

4.3 Determining Trajectories of the Generator Transition System

In this section, we will outline a procedure for finding runs of T_g satisfying an arbitrary LTL_{-X} formula ϕ over Π. Due to space constraints, we omit the details, and refer the reader to the technical report available at [28]. We start by translating ϕ into a Büchi automaton $\mathcal{B}_\phi$. To this goal, we use the conversion algorithm described in [13] and its freely downloadable implementation LTL2BA. Then we take the (synchronous) product of T_g with $\mathcal{B}_\phi$ to obtain a product automaton $\mathcal{A}_{g,\phi}$ [30]. We use standard algorithms for graph traversing on $\mathcal{A}_{g,\phi}$ and eventually project back to find the desired runs of T_g. Our approach is inspired by model checking algorithms, which are used to verify if a transition system satisfies a property expressed in terms of LTL. The difference is that a model checker constructs a Büchi automaton for the negation of the LTL formula and the product automaton is checked for emptiness (*i.e.* non-existence of accepted runs).

While we refer the reader to [28] for details, two important observations are in order. First, as opposed to related approaches reported in [30, 9], we consider possible self-transitions in states of T_g (Definition 6), and cannot use the "stutter extension" rule. In our case, the usage of this rule (adding self-loops to blocking final states of $\mathcal{A}_{g,\phi}$) could lead to incorrect results, because we could obtain runs which cannot be produced by T_g. Second, we consider only runs of T_g that have a special structure composed of one *prefix* and an infinite number of repetitions of a *suffix*. Note that this is not restrictive, since it can be proved [30] that, if there is an accepted run, then there is at least one accepted run with the above structure. If there are more such runs starting from the same state, we choose the "shortest" one, as defined in [28].

Let $r_i = r_i(1)r_i(2)r_i(3)\ldots$, $r_i(j) \in Q_g = \mathcal{P}$ denote the nonempty run of T_g starting from state q_i, *i.e.,* $r_i(1) = q_i$, $i \in I$, where $I \subseteq \{1, \ldots, \mathcal{M}\}$ is the set of indices of all nonempty runs. The fact that r_i has the prefix-suffix structure can be formally written as: for any $i \in I$, there exists n_p^i and n_s^i such that for any $j > n_p^i + n_s^i$, $r_i(j) = r_i((j - n_p^i - 1) \bmod n_s^i + n_p^i + 1)$. n_p^i and n_s^i are the number of states in prefix and suffix of r_i, respectively and thus the run r_i contains at most $n_p^i + n_s^i$ different states.

Proposition 1, proved in [28], states that, in a run r_i, $i \in I$ of T_g, none of the states can be succeeded by itself, except for the state of a suffix of length one (case in which this state will be infinitely repeated).

Proposition 1. *Each run $r_i = r_i(1)r_i(2)r_i(3)\ldots$, $i \in I$, satisfies the following property: $r_i(j) \neq r_i(j+1)$, $\forall j \in \mathbb{N}^*$, $j \neq n_p^i + k\, n_s^i + 1$, $k \in \mathbb{N}$. Moreover, if $n_s^i \geq 2$, $r_i(j) \neq r_i(j+1)$, $\forall j \in \mathbb{N}^*$.*

Remark 3. Proposition (1) and Remark 2 justifies our choice of LTL without the 'next' operator. Indeed, we do not need the increased expressiveness obtained by adding it.

5 Control Strategy

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

$$x(0) \in \cup_{i \in I} h^{-1}(q_i), \tag{9}$$

where $I \subseteq \{1, \ldots, \mathcal{M}\}$ is the set of indices of non-empty runs as defined in the previous section.

Definition 7 (Control strategy). *A control strategy for system (3) corresponding to an LTL_{-X} formula ϕ is a tuple $C^\phi = (L, L_0, u, Inv, Rel)$, where:*

- $L = \{l^i_{r_i(j)r_i(j+1)} \,|\, i \in I,\, j \geq 1\}$ *is its set of locations,*
- $L_0 = \{l^i_{q_i r_i(2)},\, i \in I\}$ *is the set of initial locations,*
- $Inv : L \rightarrow 2^{P_N}$, $Inv(l^i_{r_i(j)r_i(j+1)}) = \overline{h^{-1}(r_i(j))}$ *gives the invariant for each location,*
- $u : L \times P_N \rightarrow U$ *is a map which 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 feedback controller $u(l^i_{r_i(j)r_i(j+1)}, x) = u_{r_i(j)r_i(j+1)}(x)$ ($u_{r_i(j)r_i(j+1)}$ are defined in Section 4.2),*
- $Rel \subseteq L \times L$, $Rel = \{(l^i_{r_i(j)r_i(j+1)}, l^i_{r_i(j+1)r_i(j+2)}),\, i \in I,\, j \geq 1,\, r_i(j) \neq r_i(j+1)\}$

A location $l^i_{r_i(j)r_i(j+1)}$ corresponds to position j in run r_i. According to structure of runs described in Section 4.3, the set of locations L is finite, even though the runs are infinite. A location $l^i_{r_i(j)r_i(j+1)}$ corresponds to driving all states from $\overline{h^{-1}(r_i(j))}$ to $h^{-1}(r_i(j+1))$ 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 7 applied to system (3) with initial states (9) 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^ϕ is updated to l' according to $(l, l') \in Rel$ and the process continues.

Remark 4. 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 2 and 1, 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 7 and set of initial states as in (9), satisfy the LTL_{-X} formula ϕ and stay inside P_N for all times.*

Proof. The proof follows from the construction of C^ϕ from Definition 7, 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. The details can be found in [28].

Remark 5. 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 controllers are in general time-variant. The feedback controllers will

be piece-wise affine and with a thin set of discontinuities - the common facets of full dimensional subpolytopes of P_N. The generated trajectories will be piecewise smooth and everywhere continuous.

To implement the control strategy described in Definition 7, we have in general infinitely many choices of controllers of the type $u_{q_iq_i}$ and $u_{q_iq_j}$. Indeed, for any polytope, Corollaries 2 and 1 return whole polyhedral sets of allowed controls at vertices. In order to construct a controller according to (7), (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 $u_{q_iq_j}$ 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 $u_{q_iq_i}$ 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 $h^{-1}(q_i)$.

Discussion. Our approach to solving Problem 1 is obviously conservative. If the model checking 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. 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 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 [12], 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_N 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 [24]. 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. Due to space constraints, we do not give here an analysis of complexity. However, an example is included at the end of Section 6 for illustration.

6 Implementation and Simulation Results

We implemented our approach as a user friendly software package for LTL_{-X} control of linear systems LTLCon under Matlab. The tool, which is freely downloadable from [14], takes as input the polytope P_N, the matrices A, B, and b of system (3)), and the LTL_{-X} formula ϕ. 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 [31] and it is used to convert a polytope expressed in form (1) to form (2) and vice-versa. The second one is LTL2BA [13], which is used to convert an LTL formula to a Büchi automaton.

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

$$\dot{x} = \begin{bmatrix} 0.2 & -0.3 \\ 0.5 & -0.5 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad x \in P_2, \; u \in U \tag{10}$$

Polytope P_2 is specified in form (2), as the intersection of 8 closed half spaces, defined by: $a_1 = [-1 \quad 0]^T$, $b_1 = -5$, $a_2 = [1 \quad 0]^T$, $b_2 = -7$, $a_3 = [0 \quad -1]^T$, $b_3 = -3$, $a_4 = [0 \quad 1]^T$, $b_4 = -6$, $a_5 = [-3 \quad -5]^T$, $b_5 = -15$, $a_6 = [1 \quad -1]^T$, $b_6 = -7$, $a_7 = [-1 \quad 2.5]^T$, $b_7 = -15$, $a_8 = [-2 \quad 2.5]^T$, $b_8 = -17.5$. Control constraints are captured by the set $U = [-2, 2] \times [-2, 2]$.

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

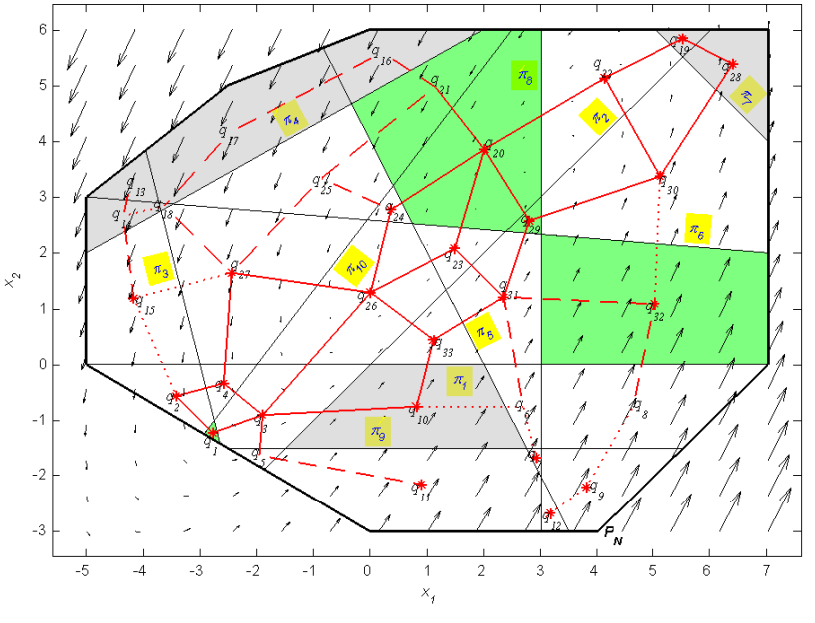


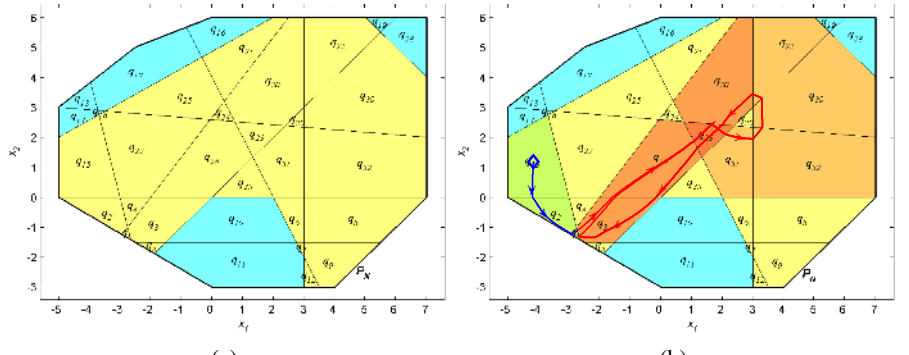
Fig. 1. The arrows represent the drift vector field of system (10). The yellow boxes mark the half-spaces corresponding to atomic propositions π_i, $i = 1, \ldots, 10$. The regions to be visited are green, while the obstacles are gray.

There are 33 feasible full-dimensional subpolytopes in P_2, and therefore 33 states in T_g. Figure 1 depicts the bounding polytope P_2, the vector field given by the drift of system (10), the predicates π_i, $i = 1, \ldots, 10$, and the feasible subpolytopes corresponding to states q_i, $i = 1, \ldots, 33$ of T_g. The red lines connecting the centroids of the polytopes in Figure 1 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 red 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 Figure 1. The LTL_{-X} formula can be written as $\phi = \Box(\diamond(r_1 \wedge \diamond(r_2 \wedge \diamond r_3)) \wedge \neg(o_1 \vee o_2 \vee o_3))$. By expressing interesting regions r_i and o_i, $i = 1, 2, 3$ in terms of predicates π_j, $j = 1, \ldots, 10$ we obtain $\phi = \Box(\diamond((\pi_3 \wedge \pi_{10}) \wedge \diamond ((\neg\pi_4 \wedge \pi_5 \wedge \pi_6 \wedge \pi_8) \wedge \diamond (\neg\pi_1 \wedge \neg\pi_6 \wedge \neg\pi_8)))$ $\wedge\neg(\pi_4 \vee \pi_7 \vee (\pi_1 \wedge \neg\pi_2 \wedge \neg\pi_5 \wedge \pi_9)))$.

The set of initial states from which there exist continuous trajectories satisfying the formula is the union of the yellow polytopes in Figure 2 (a). The set of initial states of T_g from which there exist runs satisfying the formula are the corresponding labels. The run r_{15} of T_g starting from q_{15} and satisfying ϕ is presented in Figure 2 (b). The prefix of run r_{15} of T_g is $q_{15}q_2$ (shown as green polytopes), while the suffix is $q_1q_3q_{26}q_{23}q_{20}q_{23}q_{31}q_{32}q_{30}q_{22}q_{20}q_{23}q_{26}q_3$ (red polytopes). A continuous trajectory starting from $x_0 = [-2.66 \quad -1.33]^T$ (blue diamond) is also shown in Figure 2 (b). It is colored in blue for prefix part and in red for suffix part.

The above 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



(a) (b)

Fig. 2. (a) The union of the yellow polytopes represents the set of initial states from which there exist continuous trajectories satisfying formula ϕ. (b) An example of such a trajectory.

about 0.9 seconds. The Büchi automaton had 9 states and was created in 2.2 seconds. The desired runs of T_g were obtained in about 11 seconds. We also ran a four dimensional example ($N = 4$), with P_4 defined by 9 hyperplanes and Π containing $n = 15$ predicates. There were $\mathcal{M} = 295$ states in T_g - its construction took 68 seconds. 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.

7 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 modelling and temporal logic for specifying performance. Future directions of research include the extension of these techniques to piece-wise affine systems and hybrid systems with more complicated dynamics.

References

1. Emerson, E.A.: Temporal and modal logic. In van Leeuwen, J., ed.: Handbook of Theoretical Computer Science: Formal Models and Semantics. Volume B. North-Holland Pub. Co./MIT Press (1990) 995–1072
2. Shults, B., Kuipers, B.: Proving properties of continuous systems: Qualitative simulation and temporal logic. Artificial Intelligence **92** (1997) 91–130
3. Brajnik, G., Clancy, D.: Focusing qualitative simulation using temporal logic: theoretical foundations. Annals of Mathematics and Artificial Intelligence **22** (1998) 59–86
4. Davoren, J., Coulthard, V., Markey, N., Moor, T.: Non-deterministic temporal logics for general flow systems. In: The 7th International Workshop on Hybrid Systems: Computation and Control, Philadelphia, PA (2004) 280–295
5. Tabuada, P., Pappas, G.: Model checking ltl over controllable linear systems is decidable. Volume 2623 of Lecture Notes in Computer Science. Springer-Verlag (2003)
6. Antoniotti, M., Mishra, B.: Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. In: IEEE International Conference on Robotics and Automation. (1995)
7. Loizou, S.G., Kyriakopoulos, K.J.: Automatic synthesis of multiagent motion tasks based on ltl specifications. In: 43rd IEEE Conference on Decision and Control. (2004)
8. Quottrup, M.M., Bak, T., Izadi-Zamanabadi, R.: Multi-robot motion planning: A timed automata approach. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA (2004) 44174422
9. Fainekos, G.E., Kress-Gazit, H., Pappas, G.J.: Hybrid controllers for path planning: a temporal logic approach. In: Proceedings of the 2005 IEEE Conference on Decision and Control, Seville, Spain (2005)
10. Antoniotti, M., Park, F., Policriti, A., Ugel, N., Mishra, B.: Foundations of a query and simulation system for the modeling of biochemical and biological processes. Proceedings of the Pacific Symposium on Biocomputing, Lihue, Hawaii (2003) 116–127

11. Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M., Schneider, D.: Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in e.coli. In: Thirteen International Conference on Intelligent Systems for Molecular Biology. (2005)
12. Habets, L., van Schuppen, J.: A control problem for affine dynamical systems on a full-dimensional polytope. Automatica **40** (2004) 21–35
13. Gastin, P., Oddoux, D.: Fast ltl to büchi automata translation. In G. Berry, H.C., Finkel, A., eds.: Proceedings of the 13th Conference on Computer Aided Verification (CAV'01). Number 2102 in LNCS, SPRINGER (2001) 53–65
14. Kloetzer, M., Belta, C.: Ltlcon, a matlab package for control of linear systems from linear temporal logic specifications. (2005) URL http//iasi.bu.edu/∼software/LTL-control.htm.
15. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
16. Alur, R., C.Courcoubetis, Halbwachs, N., Henzinger, T., Ho, P., Nicollin, X., A.Olivero, Sifakis, J., Yovine, S.: Hybrid automata: An algorithmic approach to specification and verification of hybrid systems. Theoretical Computer Science **138** (1995) 3–34
17. Puri, A., Varaiya, P.: Decidability of hybrid systems with rectangular inclusions. Computer Aided Verification (1994) 95–104
18. Lafferriere, G., Pappas, G.J., Sastry, S.: O-minimal hybrid systems. Mathematics of Control, Signals and Systems **13** (2000) 1–21
19. Alur, R., Henzinger, T.A., Lafferriere, G., Pappas, G.J.: Discrete abstractions of hybrid systems. Proceedings of the IEEE **88** (2000) 971–984
20. Pappas, G.J.: Bisimilar linear systems. Automatica **39** (2003) 2035–2047
21. Broucke, M.: A geometric approach to bisimulation and verification of hybrid systems. In Vaandrager, F.W., van Schuppen, J.H., eds.: Hybrid Systems: Computation and Control. Volume 1569 of Lecture Notes in Computer Science. Springer-Verlag (1999) 61–75
22. Haghverdi, E., Tabuada, P., Pappas, G.: Bisimulation relations for dynamical and control systems. Volume 69 of Electronic Notes in Theoretical Computer Science. Elsevier (2003)
23. Tiwari, A., Khanna, G.: Series of abstractions for hybrid automata. In: Fifth International Workshop on Hybrid Systems: Computation and Control, Stanford, CA (2002)
24. Belta, C., Isler, V., Pappas, G.J.: Discrete abstractions for robot planning and control in polygonal environments. IEEE Transactions on Robotics **21** (2005) 864–874
25. Motzkin, T., H.Raiffa, Thompson, G., R.M.Thrall: The double description method. In Kuhn, H., Tucker, A., eds.: Contributions to theory of games. Volume 2. Princeton University Press, Princeton, NJ (1953)
26. Fukuda, K.: cdd/cdd+ package. (URL http://www.cs.mcgill.ca/∼fukuda/soft/cdd_home/cdd.html)
27. Lee, C.W.: Subdivisions and triangulations of polytopes. In Goodman, J.E., ORourke, J., eds.: Handbook of discrete and computational geometry. CRC Press, Boca Raton, NY (1997) 271–290
28. Kloetzer, M., Belta, C.: A fully automated framework for controller synthesis from linear temporal logic specifications. Technical Report CISE 2005-IR-0050, Boston University (2005) http://www.bu.edu/systems/research/publications/2005/2005-IR-0050.pdf.
29. Belta, C., Habets, L.: Constructing decidable hybrid systems with velocity bounds. In: 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas (2004)
30. Holzmann, G.: The Spin Model Checker, Primer and Reference Manual. Addison-Wesley, Reading, Massachusetts (2004)
31. Torrisi, F., Baotic, M.: Matlab interface for the cdd solver. (URL http:// control.ee.ethz.ch/∼hybrid/cdd.php)