Language-Guided Controller Synthesis for Discrete-Time Linear Systems

Ebru Aydin Gol
Boston University
Boston, MA 02215, USA
ebru@bu.edu

Mircea Lazar
Eindhoven University of Technology
Den Dolech 2, 5600MB,
Eindhoven, The Netherlands.
m.lazar@tue.nl

Calin Belta
Boston University
Boston, MA 02215, USA
cbelta@bu.edu

ABSTRACT

This paper considers the problem of controlling discrete-time linear systems from specifications given as formulas of syntactically co-safe linear temporal logic over linear predicates in the state variables of the system. A systematic procedure is developed for the automatic computation of sets of initial states and feedback controllers such that all the resulting trajectories of the corresponding closed-loop system satisfy the given specifications. The procedure is based on the iterative construction and refinement of an automaton that enforces the satisfaction of the formula. Interpolation and polyhedral Lyapunov function based approaches are proposed to compute the polytope-to-polytope controllers that label the transitions of the automaton. The algorithms developed in this paper were implemented as a software package that is available for download. Their application and effectiveness are demonstrated for two challenging case studies.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Control theory*; D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal methods*

Keywords

Linear temporal logic, Automata theory, Constrained control, Polytope-to-polytope control, Polyhedral Lyapunov functions

1. INTRODUCTION

Temporal logics, such as linear temporal logic (LTL) and computation tree logic (CTL), and model checking algorithms [1] have been primarily used for specifying and verifying the correctness of software and hardware systems. In recent years, due to their expressivity and resemblance to natural language, temporal logics have gained increasing popularity as specification languages in other areas such as dynamical systems [2–6], biology [7–9], and robotics [10–14]. These new application areas also have emphasized the need for formal synthesis, where the goal is to generate a control strategy for a dynamical system from a specification given as a temporal logic formula. Recent efforts resulted in control algorithms for continuous and discrete-time linear systems from specifications given as LTL formulas [3, 6], motion planning and control strategies for robotic systems from specifications given in μ-calculus [11], CTL [12], LTL [13], and fragments of LTL such as GR(1) [5, 10] and syntactically co-safe LTL [14].

In this paper, we consider the following problem: given a discrete-time linear system and a syntactically co-safe LTL formula [15] over linear predicates in the states of the system, find a set of initial states, if possible, the largest, for which there exists a control strategy such that all the trajectories of the closed-loop system satisfy the formula. The syntactically co-safe fragment of LTL is rich enough to express a wide spectrum of finite-time properties of dynamical systems, such as finite-time reachability of a target with obstacle avoidance ("go to A and avoid B and C for all times before reaching A"), enabling conditions ("do not go to D unless E was visited before"), and temporal logic combinations of the above. For example, the syntactically co-safe LTL formula "$(\neg O \mathscr{U} T) \wedge (\neg T \mathscr{U} (R_1 \vee R_2))$" requires convergence to target region T through regions R_1 or R_2 while avoiding obstacle O.

Central to our "language-guided" approach to the above problem is the construction and refinement of an automaton that restricts the search for initial states and control strategies in such a way that the satisfaction of the specifications is guaranteed at all times. The states of the automaton correspond to polytopic subsets of the state-space. Its transitions are labeled by state-feedback controllers that drive the states of the original system from one polytope to another. We propose techniques based on vertex interpolation and polyhedral Lyapunov functions (LFs) for the construction of these controllers. The refinement procedure iteratively partitions the state regions, modifies the automaton, and updates the set of initial satisfying states by performing a search and a backward reachability analysis on the graph of the automaton. The automaton obtained at the end of the iteration process provides a control strategy that solves the initial problem.

The contribution of this work is twofold. First, we provide a computational framework in which the exploration of the state-space is "guided" by the specification. This is in contrast with existing related works [6, 16], in which an abstraction is first constructed through the design of polytope-to-polytope feedback controllers, and then controlled by solving a temporal logic game on the abstraction. By combining the abstraction and the automaton control processes, the method proposed in this paper avoids regions of the state-space that do not contain satisfying initial states, and is, as a result, more efficient. In addition, it naturally induces an iterative refinement and enlargement of the set of initial conditions, which was not possible in [16] and was not formula-guided in [6].

HSCCÕ12, April 17–19, 2012, Beijing, China.

Second, this paper provides an extension of previous results on obstacle avoidance [17–19] in certain directions. For example, it provides a systematic way to explore the feasible state-space from "rich" temporal logic specifications that are not limited to going to a target while avoiding a set of obstacles. Furthermore, it does not necessarily involve paths characterized by unions of overlapping polytopes and the existence of artificial closed-loop equilibria. Also, as a byproduct, the approach developed in this paper provides an upper bound for the time necessary to satisfy the temporal logic specifications by all the trajectories originating from the constructed set of initial states.

The remainder of the paper is organized as follows. We review some notions necessary throughout the paper in Sec. 2 before formulating the problem and outlining the approach in Sec. 3. The iterative construction of the abstraction is presented in Sec. 4. The LP-based algorithms for solving polytope-to-polytope control problems are described in Sec. 5. The main theorem is stated in Sec. 6, while illustrative examples are shown in Sec. 7. Conclusions are summarized in Sec. 8.

2. NOTATION AND PRELIMINARIES

In this section, we introduce the notation and provide some background on temporal logic and automata theory. For a set $\mathscr{S}$, $\text{int}(\mathscr{S})$, $\text{Co}(\mathscr{S})$, $\#\mathscr{S}$, and $2^{\mathscr{S}}$ stand for its interior, convex hull, cardinality, and power set, respectively. For $\lambda \in \mathbb{R}$ and $\mathscr{S} \subset \mathbb{R}^n$, let $\lambda\mathscr{S} := \{\lambda x | x \in \mathscr{S}\}$. We use $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$, and $\mathbb{Z}_+$ to denote the sets of real numbers, non-negative reals, integer numbers, and non-negative integers. For $m, n \in \mathbb{Z}_+$, we use $\mathbb{R}^n$ and $\mathbb{R}^{m\times n}$ to denote the set of column vectors and matrices with n and $m \times n$ real entries. $I_n \in \mathbb{R}^{n\times n}$ stands for the $n \times n$ identity matrix. For a matrix A, $A_{i\bullet}$ and $A_{\bullet j}$ denote its i-th row and j-th column, respectively. Given a vector $x \in \mathbb{R}^n$, $\|x\|$ denotes its p-norm (the value of p will be clear from the context).

A polyhedron (polyhedral set) in $\mathbb{R}^n$ is the intersection of a finite number of open and/or closed half-spaces. A polytope is a compact polyhedron. We use $\mathscr{V}(\mathscr{P})$ to denote the set of vertices of a polytope $\mathscr{P}$. Both the $\mathscr{V}$-representation ($\text{Co}(\mathscr{V}(\mathscr{P}))$) and the $\mathscr{H}$-representation ($\{x \in \mathbb{R}^n \mid H_{\mathscr{P}} x \leq h_{\mathscr{P}}\}$, where matrix $H_{\mathscr{P}}$ and vector $h_{\mathscr{P}}$ have suitable dimensions) [20] of a polytope $\mathscr{P}$ will be used throughout the paper.

In this work, the control specifications are given as formulas of syntactically co-safe linear temporal logic (scLTL).

Definition 2.1 [21] A scLTL formula over a set of atomic propositions P is inductively defined as follows:

$$\Phi := p | \neg p | \Phi \vee \Phi | \Phi \wedge \Phi | \Phi \mathscr{U} \Phi | \bigcirc \Phi | \Diamond \Phi, \tag{1}$$

where p is an atomic proposition, $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction) are Boolean operators, and $\bigcirc$ ("next"), $\mathscr{U}$ ("until"), and $\Diamond$ ("eventually") are temporal operators.

The semantics of scLTL formulas is defined over infinite words over 2^P as follows:

Definition 2.2 The satisfaction of a scLTL formula Φ at position $i \in \mathbb{Z}_+$ of a word w over 2^P, denoted by $w_i \models \Phi$, is recursively defined as follows: 1) $w_i \models p$ if $p \in w_i$, 2) $w_i \models \neg p$ if $p \notin w_i$, 3) $w_i \models \Phi_1 \vee \Phi_2$ if $w_i \models \Phi_1$ or $w_i \models \Phi_2$, 4) $w_i \models \bigcirc \Phi$ if $w_{i+1} \models \Phi$, 5) $w_i \models \Phi_1 \mathscr{U} \Phi_2$ if there exists $j \geq i$ such that $w_j \models \Phi_2$ and for all $i \leq k < j$ $w_k \models \Phi_1$, and 6) $w_i \models \Diamond \Phi$ if there exists $j \geq i$ such that $w_j \models \Phi$.

A word w satisfies a scLTL formula Φ, written as $w \models \Phi$, if $w_0 \models \Phi$.

An important property of scLTL formulas is that, even though they have infinite-time semantics, their satisfaction is guaranteed in finite time. Explicitly, for any scLTL formula Φ over P, any satisfying infinite word over 2^P contains a satisfying finite prefix. We use $\mathscr{L}_\Phi$ to denote the set of all (finite) prefixes of all satisfying infinite words.

Definition 2.3 A deterministic finite state automaton (FSA) is a tuple $\mathscr{A} = (Q, \Sigma, \rightarrow_{\mathscr{A}}, Q_0, F)$ where Q is a finite set of states, Σ is a set of symbols, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states and $\rightarrow_{\mathscr{A}} \subseteq Q \times \Sigma \times Q$ is a deterministic transition relation.

An accepting run $r_{\mathscr{A}}$ of an automaton $\mathscr{A}$ on a finite word $w = w_0 w_1 \dots w_d$ over Σ is a sequence of states $r_{\mathscr{A}} = q_0 q_1 \dots q_{d+1}$ such that $q_0 \in Q_0$, $q_{d+1} \in F$ and $(q_i, w_i, q_{i+1}) \in \rightarrow_{\mathscr{A}}$ for all $i = 0, \dots, d$. The set of all words corresponding to all of the accepting runs of $\mathscr{A}$ is called the language accepted by $\mathscr{A}$ and is denoted as $\mathscr{L}_{\mathscr{A}}$.

For any scLTL Φ formula over P, there exists a FSA $\mathscr{A}$ with input alphabet 2^P that accepts the prefixes of all the satisfying words, i.e., $\mathscr{L}_\Phi$ [21]. There are algorithmic procedures and off-the-shelf tools, such as *scheck2* [22], for the construction of such an automaton.

Definition 2.4 A finite state generator automaton is a tuple $\mathscr{A} = (Q, \rightarrow_{\mathscr{A}}, \Gamma, \tau, Q_0, F)$ where Q is a finite set of states, $\rightarrow_{\mathscr{A}} \subseteq Q \times Q$ is a non-deterministic transition relation, Γ is a set of output symbols, $\tau : Q \rightarrow \Gamma$ is an output function, $Q_0 \subseteq Q$ is a set of initial states and $F \subseteq Q$ is a set of final states.

An accepting run $r_{\mathscr{A}}$ of a finite state generator automaton is a sequence of states $r_{\mathscr{A}} = q_0 q_1 \dots q_d$ such that $q_0 \in Q_0$, $q_d \in F$ and $(q_i, q_{i+1}) \in \rightarrow_{\mathscr{A}}$ for all $i = 0, \dots, d-1$. An accepting run $r_{\mathscr{A}}$ produces a word $w = w_0 w_1 \dots w_d$ over Γ such that $\tau(q_i) = w_i$, for all $i = 0, \dots, d$. The output language $\mathscr{L}_{\mathscr{A}}$ of a finite state generator automaton $\mathscr{A}$ is the set of all words that are generated by accepting runs of $\mathscr{A}$.

3. PROBLEM FORMULATION

Consider a discrete-time linear control system of the form

$$x_{k+1} = Ax_k + Bu_k, \quad x_k \in \mathbb{X},\ u_k \in \mathbb{U}, \tag{2}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ describe the system dynamics and $x_k \in \mathbb{X} \subset \mathbb{R}^n$ and $u_k \in \mathbb{U} \subset \mathbb{R}^m$ are the state and applied control at time $k \in \mathbb{Z}_+$, respectively.

Let $P = \{p_i\}_{i=0,\dots,l}$ for some $l \geq 1$ be a set of atomic propositions given as linear inequalities in $\mathbb{R}^n$. Each atomic proposition p_i induces a half-space

$$[p_i] := \{x \in \mathbb{R}^n \mid c_i^\top x + d_i \leq 0\},\ c_i \in \mathbb{R}^n, d_i \in \mathbb{R}. \tag{3}$$

A trajectory $x_0 x_1 \dots$ of system (2) produces a word $P_0 P_1 \dots$ where $P_i \subseteq P$ is the set of atomic propositions satisfied by x_i, i.e., $P_i = \{p_j \mid \exists j \in \{0, \dots, l\}, x_i \in [p_j]\}$. The specifications are given as scLTL formulas over the set of predicates P. A system trajectory satisfies a specification if the word produced by the trajectory satisfies the corresponding formula. The main problem considered in this paper can be formulated as follows:

Problem 3.1 Given a scLTL formula Φ over a set of linear predicates P and a dynamical system as defined in Eqn. (2), construct a set of initial states $\mathbb{X}_0$ and a feedback control strategy such that all the words produced by the closed-loop trajectories originating in $\mathbb{X}_0$ satisfy formula Φ.

We propose a solution to the above problem by relating the control synthesis problem with a finite state generator automaton (Def. (2.4)), whose states correspond to polyhedral subsets of the system state-space and whose transitions are mapped to state feedback controllers. This automaton will be constructed as the dual of the automaton that accepts the language satisfying formula Φ. Its states will be refined until feasible polytope-to-polytope control problems are obtained. This approach reduces the controller synthesis part of Prob. 3.1 to solving a finite number of polytope-to-polytope control problems.

The proposed solutions to polytope-to-polytope controller synthesis will supply a worst case time bound such that every trajectory originating from the source polytope reaches the target polytope within the provided time bound. These bounds can be further used to compute an upper time bound for a given initial state, such that the trajectory starting from this state satisfies the specification within the computed time bound.

4. AUTOMATON GENERATION AND REFINEMENT

In this section, we present algorithms for the construction and refinement of the dual automaton that corresponds to a desired set of LTL specifications.

4.1 FSA and dual automaton

All words that satisfy the specification formula Φ are accepted by a FSA $\mathscr{A} = (Q, 2^P, \rightarrow_{\mathscr{A}}, Q_0, F)$. The dual automaton $\mathscr{A}^D = (Q^D, \rightarrow^D, \Gamma^D, \tau^D, Q_0^D, F^D)$ is constructed as a finite state generator automaton by interchanging the states and the transitions of the automaton $\mathscr{A}$. As the transitions of $\mathscr{A}$ become states of $\mathscr{A}^D$, elements from 2^P label the states and define polyhedral sets within the state-space of system (2).

Definition 4.1 Given a FSA $\mathscr{A} = (Q, \Sigma, \rightarrow_{\mathscr{A}}, Q_0, F)$, its dual automaton is a tuple $\mathscr{A}^D = (Q^D, \rightarrow^D, \Gamma^D, \tau^D, Q_0^D, F^D)$ where

$$
\begin{aligned}
Q^D &= \{(q,\sigma,q') \mid (q,\sigma,q') \in \rightarrow_{\mathscr{A}}\},\\
\rightarrow^D &= \{((q,\sigma,q'),(q',\sigma',q'')) \mid (q,\sigma,q'),(q',\sigma,q'') \in \rightarrow_{\mathscr{A}}\},\\
\Gamma^D &= 2^P,\\
\tau^D &: Q^D \rightarrow \Gamma^D, \quad \tau^D((q,\sigma,q')) = \sigma,\\
Q_0^D &= \{(q_0,\sigma,q) \mid q_0 \in Q_0\},\\
F^D &= \{(q,\sigma,q') \mid q' \in F\}.
\end{aligned}
$$

Informally, the states of the dual automaton $\mathscr{A}^D$ are the transitions of the automaton $\mathscr{A}$. A transition is defined between two states of $\mathscr{A}^D$ if the corresponding transitions are connected by a state in $\mathscr{A}$. The set of output symbols of $\mathscr{A}^D$ is the same as the set of symbols of $\mathscr{A}$. For a state of $\mathscr{A}^D$, the output function produces the symbol that enables the transition in $\mathscr{A}$. The set of initial states Q_0^D of $\mathscr{A}^D$ is the set of all transitions that leave an initial state in $\mathscr{A}$. Similarly, the set of final states F^D of $\mathscr{A}^D$ is the set of transitions that end in a final state of $\mathscr{A}$. The construction of $\mathscr{A}^D$ guarantees that any word produced by $\mathscr{A}^D$ is accepted by $\mathscr{A}$:

Proposition 4.2 *The output language of the dual automaton $\mathscr{A}^D$ coincides with the language accepted by the automaton $\mathscr{A}$, i.e., $\mathscr{L}_{\mathscr{A}} = \mathscr{L}_{\mathscr{A}^D}$.*

The proof of Prop. 4.2 follows directly from the definitions of the automata and is omitted for brevity.

4.1.1 Automaton Representation

A FSA $\mathscr{A}$ that accepts the language of a scLTL formula Φ over P is constructed with the tool *scheck2* [22]. This tool labels each transition of the produced FSA with a disjunctive normal form (DNF) $C_1 \vee C_2 \vee \ldots \vee C_d$, where each C_i is a conjunctive clause over P. This is a compact representation of the corresponding FSA in which each transition is labeled by a conjunctive clause.

In what follows, we use $\mathscr{P}_q \subseteq \mathbb{X}$ to denote the set of states of system (2) that satisfy the Boolean formula of a dual automaton state q. Given a DNF formula $D = C_1 \vee C_2 \vee \ldots \vee C_d$, $\mathscr{P}_{C_i} := [p_{i_1}] \cap \ldots \cap [p_{i_c}]$ denotes the set of states of system (2) that satisfy $C_i = p_{i_1} \wedge \ldots \wedge p_{i_c}$ where $i_j \in 0,\ldots,l, \forall j \in 1,\ldots,c$ and $\mathscr{P}_D := \cup_{i=1}^{d} \mathscr{P}_{C_i}$ denotes the set of states of system (2) that satisfy D.

While constructing the dual automaton, each of the conjunctive clauses is used as a separate transition, which ensures that all corresponding subsets of the state-space are polyhedra. Before constructing the dual automaton each DNF formula $C_1 \vee C_2 \vee \ldots \vee C_d$ is simplified by applying the following rules:

- *Empty set elimination:* C_i is eliminated if the corresponding region is empty, i.e., $\mathscr{P}_{C_i} = \emptyset$. The symbols that satisfy such clauses can not be generated by the system trajectories.
- *Subset elimination:* C_i is eliminated if its corresponding set is a subset of the set corresponding to C_j, $j \neq i$, i.e., $\mathscr{P}_{C_i} \subseteq \mathscr{P}_{C_j}$. The system states that satisfy C_i also satisfy C_j which enables the same transition.

Even though these simplifications change the language of the dual automaton, it can be easily seen that the set of corresponding satisfying trajectories of system (2) is preserved.

Example 4.3 A simple example is used to explain the construction routines. Consider the following scLTL formula:

$$\Phi_1 = (p_0 \wedge p_1 \wedge p_2) \; \mathscr{U} \; (p_1 \wedge p_2 \wedge p_3 \wedge p_4) \tag{4}$$

over $P = \{p_0, p_1, p_2, p_3, p_4\}$, where $c_0 = [-1,1]^\top$, $d_0 = 0$, $c_1 = [1,1]^\top$, $d_1 = 4$, $c_2 = [0,1]^\top$, $d_2 = -0.1$, $c_3 = [-1,0]^\top$, $d_3 = -3$, $c_4 = [1,0]^\top$, $d_4 = 5$. The trajectories that satisfy Φ_1 evolve in the region $[p_0] \cap [p_1] \cap [p_2]$ until they reach the target region $[p_1] \cap [p_2] \cap [p_3] \cap [p_4]$. The regions defined by this set of predicates are given in Fig. 1. The compact representation of a FSA that accepts the language satisfying formula Φ_1 is shown in Fig. 2. For example, the transition from the state labeled with "0" to the state labeled with "1", which is labeled by $(p_4 \wedge p_3 \wedge p_2 \wedge p_1)$, corresponds to two transitions labeled by $\{p_0, p_1, p_2, p_3, p_4\}$ and $\{p_1, p_2, p_3, p_4\}$, respectively.

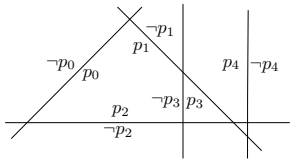


Figure 1: Half-spaces generated by the linear predicates in Eqn. (4)**.**

The compact representations of dual automata constructed with and without simplifying the DNF formulas are shown in Fig. 3, where a state label corresponds to the subsets of 2^P which can be produced by τ^D in that state. The simplification deletes $(\neg p_4 \wedge p_2 \wedge p_1 \wedge p_0)$ from the self transition of the state labeled with "0" in Fig. 2, since the set of states that satisfies this clause is empty.

An accepting run $r_D = q_0 q_1 \ldots q_d$ of $\mathscr{A}^D$ defines a sequence of polyhedral sets $\mathscr{P}_{q_0} \mathscr{P}_{q_1} \ldots \mathscr{P}_{q_d}$. Any trajectory $x_0 x_1 \ldots x_d$ of the

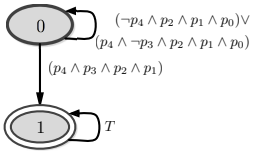


Figure 2: Compact representation of a FSA that accepts the language satisfying formula Φ_1 in Eqn. (4). The initial states are filled with grey and the final state is marked with a double circle.

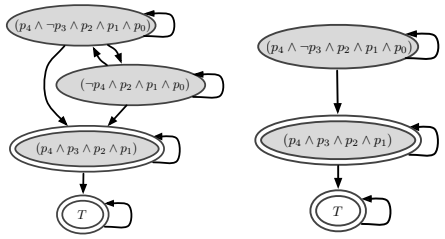


Figure 3: Dual automata for the FSA from Fig. 2: (a) without Boolean simplification; (b) with Boolean simplification. T stands for the Boolean constant true.

original system (2) with $x_i \in \mathscr{P}_{q_i}$, $i = 0,\ldots,d$ satisfies the specification by Prop. 4.2.

We say that a transition (q,q') of $\mathscr{A}^D$ is *enabled* if there exists an admissible control law that achieves the transition for all $x \in \mathscr{P}_q$. Two conditions are introduced for constructing admissible controllers according to existence of a self transition of the source state q. When $(q,q) \in \rightarrow^D$, a controller *enables* a transition (q,q') if the corresponding closed-loop trajectories originating in $\mathscr{P}_q$ reach $\mathscr{P}_{q'}$ in finite time and remain within $\mathscr{P}_q$ until they reach $\mathscr{P}_{q'}$. When $(q,q) \notin \rightarrow^D$, a transition (q,q') is only enabled if there exists a controller such that the resulting closed-loop trajectory originating in $\mathscr{P}_q$ reaches $\mathscr{P}_{q'}$ at the next discrete-time instant. For every transition of $\mathscr{A}^D$, if a controller that enables the transition can be constructed, then every resulting closed-loop trajectory originating in $\cup_{q_0 \in Q_0^D} \mathscr{P}_{q_0}$ will satisfy the specifications by Prop. 4.2. However, existence of such controllers is not guaranteed for all the states of system (2) within $\mathbb{X}$.

Prob. 3.1 aims at finding a subset of $\mathbb{X}$ for which the polytope-to-polytope control problems induced by scLTL specifications are feasible. To this end, first, the dual automaton is pruned by checking the feasibility of transitions and states for the given system (2). Second, an iterative partitioning procedure based on a combination of backward and forward reachability will be applied to the automaton states, which correspond to polytopic subsets of $\mathbb{X}$.

4.1.2 Initial Pruning

The feasibility of the transitions of the dual automaton is first checked by considering the particular dynamics of system (2) and the set $\mathbb{U}$ where the control input takes values. $Post(\mathscr{P})$ denotes the set of states that can be reached from $\mathscr{P}$ in one discrete-time instant under the dynamics (2). For a transition (q,q'), if $Post(\mathscr{P}_q) \cap \mathscr{P}_{q'} = \emptyset$, then this transition is considered *infeasible*, since there is no admissible controller that enables this transition. As $\mathscr{P}$ and $\mathbb{U}$ are polytopes, $Post(\mathscr{P})$ can be computed as follows:

$$Post(\mathscr{P}) = \mathrm{Co}(\{Ax+Bu \mid x \in \mathscr{V}(\mathscr{P}), u \in \mathscr{V}(\mathbb{U})\}). \qquad (5)$$

Alg. 1 summarizes the pruning procedure. Once the infeasible transitions are removed as in line 1, the following feasibility tests are performed. A state and all of its adjacent transitions are deleted either if it does not have an outgoing transition and it is not a final state or if it does not have an incoming transition and it is not an initial state (line 6). Removing such states and transitions does not reduce the solution space since such states cannot be part of any satisfying trajectory.

Algorithm 1 Initial Pruning of $\mathscr{A}^D$

1: $\rightarrow^D := \rightarrow^D \setminus \{(q,q') \mid Post(\mathscr{P}_q) \cap \mathscr{P}_{q'} = \emptyset\}$
2: $\bar{Q} := Q_0^D$
3: **while** $\bar{Q} \neq \emptyset$ **do**
4: **for all** $q \in \bar{Q}$ **do**
5: $\bar{Q} := \bar{Q} \setminus \{q\}$
6: **if** $(q \notin F^D$ AND $\{q' \mid (q,q') \in \rightarrow^D\} = \emptyset)$ OR ($q \notin Q_0^D$ AND $\{q' \mid (q',q) \in \rightarrow^D\} = \emptyset$) **then**
7: $Q^D := Q^D \setminus \{q\}$
8: $\bar{Q} := \bar{Q} \cup (\{q' \mid (q,q') \in \rightarrow^D\} \cup \{q' \mid (q',q) \in \rightarrow^D\})$
9: $\rightarrow^D := \rightarrow^D \setminus (\{(q,q') \mid (q,q') \in \rightarrow^D\} \cup \{(q',q) \mid (q',q) \in \rightarrow^D\})$
10: **end if**
11: **end for**
12: **end while**

4.2 Automaton Refinement

Alg. 1 guarantees that a non-empty polyhedral subset of a source polytope $\mathscr{P}_q$ is one-step controllable to the target polytope $\mathscr{P}_{q'}$ corresponding to the transition (q,q'). However, this does not imply the feasibility of the corresponding polytope-to-polytope control problem. An iterative algorithm is developed to refine the polytope $\mathscr{P}_q$ and hence, the corresponding state of the dual automaton, whenever the feasibility test fails. Alg. 2 refines the automaton at each iteration by partitioning the states for which there does not exist an admissible sequence of control actions with respect to reaching a final state. The algorithm does not affect the states of system (2) that can reach a final state region and as such, it results in a monotonically increasing, with respect to set inclusion, set of states of system (2) for which there exists an admissible control strategy.

For a transition $(q,q') \in \rightarrow^D$, the set of states in $\mathscr{P}_q$ that can reach $\mathscr{P}_{q'}$ in one step is called a *beacon*. We use $\mathscr{B}_{qq'}$ to denote the beacon corresponding to transition (q,q'), which can be obtained as $\mathscr{B}_{qq'} := \mathscr{P}_q \cap Pre(\mathscr{P}_{q'})$, where

$$Pre(\mathscr{P}) := \{x \in \mathbb{X} \mid \exists u \in \mathbb{U},\, Ax+Bu \in \mathscr{P}\}, \quad \forall \mathscr{P} \subseteq \mathbb{R}^n. \qquad (6)$$

If $\mathscr{P}$ and $\mathbb{U}$ are polytopes, then $Pre(\mathscr{P})$ can be computed via orthogonal projection. Given a controller that enables a transition (q,q'), the cost $J((q,q'))$ of transition (q,q') is defined as the worst-case time bound such that every trajectory originating in $\mathscr{P}_q$ reaches $\mathscr{P}_{q'}$. The cost $J^P(q)$ of a state q is defined as the shortest path cost from q to a final state on the graph of the automaton weighted with transition costs.

The refinement algorithm uses three subroutines: *ShortestPath*, *Partitioning* and $FeasibilityTest(q,q')$. The *ShortestPath* procedure computes a shortest path cost for every state of $\mathscr{A}^D$ using Dijkstra's algorithm [23]. The *Partitioning* procedure, which will be presented in detail in the next subsection, partitions a state region and modifies $\mathscr{A}^D$ accordingly.

The $FeasibilityTest(q,q')$ procedure checks if there exists a controller that enables (q,q') and returns the cost $J((q,q'))$ of the transition. The computational aspects of this procedure are presented in Sec. 5. The cost is set to infinity when no feasible controller is found. When q has a self transition, the procedure checks if there exists a controller that steers all trajectories originating in $\mathscr{P}_q$ to the beacon of (q,q'), i.e., $\mathscr{B}_{qq'}$, in finite time without leaving the set $\mathscr{P}_q$. Notice that to solve the $\mathscr{P}_q$-to-$\mathscr{P}_{q'}$ problem it suffices

to solve the $\mathscr{P}_q$-to-$\mathscr{B}_{qq'}$ problem, since a trajectory originating in $\mathscr{P}_q$ will reach $\mathscr{P}_{q'}$ without leaving $\mathscr{P}_q$ only through the beacon $\mathscr{B}_{qq'}$. By definition, there exists an admissible control action for all $x \in \mathscr{B}_{qq'}$ such that $\mathscr{P}_{q'}$ is reached in one step. If q does not have a self transition, the transition (q,q') is only enabled when $\mathscr{P}_q = \mathscr{B}_{qq'}$, since $\mathscr{B}_{qq'}$ is the largest set of states in $\mathscr{P}_q$ that can reach $\mathscr{P}_{q'}$ in one step.

Algorithm 2 Refinement of $\mathscr{A}^D$

1: **for all** $(q,q') \in \rightarrow_D$ **do**
2: $J((q,q')) := FeasibilityTest(q,q')$
3: **end for**
4: $J^P := ShortestPath(J, F^D)$
5: $CandidateSet = \{(q_i,q_j) \mid (q_i,q_j) \in \rightarrow^D, J^P(q_i) = \infty, J^P(q_j) \neq \infty\}$
6: **while** $CandidateSet \neq \emptyset$ **do**
7: $(q_s,q_d) := \min_{J^P(q_j)} \{(q_i,q_j) \mid (q_i,q_j) \in CandidateSet\}$
8: $[\mathscr{A}^D, J] := Partitioning(\mathscr{A}^D, q_s, (q_s,q_d))$
9: $J^P := ShortestPath(J, F^D)$
10: $CandidateSet := \{(q_i,q_j) \mid (q_i,q_j) \in \rightarrow^D, J^P(q_i) = \infty, J^P(q_j) \neq \infty\}$
11: **end while**

At each iteration of the refinement algorithm, the transition costs and shortest path costs are updated, and the set of candidate states for partitioning is constructed as follows. A state q_i that has an infinite cost ($J^P(q_i) = \infty$) and a transition ($(q_i,q_j) \in \rightarrow^D$) to a state that has a finite cost ($J^P(q_j) < \infty$) is chosen as a candidate state for partitioning (lines 5 and 10). Then, a state q_s is selected from the set of candidate states for partitioning by considering the path costs in line 7. The algorithm stops when there are no transitions from infinite cost states to finite cost states, i.e., when the set of candidate states for partitioning is empty.

4.2.1 Partitioning

A state q is partitioned into a set of states $\{q_1,\dots,q_d\}$ via a polytopic partition of $\mathscr{P}_q$. The transitions of the new states are inherited from the state q and new states are set as start states if $q \in Q_0^D$ to preserve the automaton language. The partitioning procedure is summarized in Alg. 3.

Algorithm 3 Partitioning of q in $\{q_1,\dots,q_d\}$

$Q^D := (Q^D \setminus \{q\}) \cup \{q_1,\dots,q_d\}$
for all $(q',q) \in \rightarrow^D$ **do**
  $\rightarrow^D := \rightarrow^D \setminus \{(q',q)\}$
  for $i = 1 : d$ **do**
    if $Post(q') \cap \mathscr{P}_{q_i} \neq \emptyset$ **then**
      $\rightarrow^D := \rightarrow^D \cup \{(q',q_i)\}$
      $J((q',q_i)) := FeasibilityTest(q',q_i)$
    end if
  end for
end for
for all $(q,q') \in \rightarrow^D$ **do**
  $\rightarrow^D := \rightarrow^D \setminus \{(q,q')\}$
  for $i = 1 : d$ **do**
    if $Post(q_i) \cap \mathscr{P}_{q'} \neq \emptyset$ **then**
      $\rightarrow^D := \rightarrow^D \cup \{(q_i,q')\}$
      $J((q_i,q')) := FeasibilityTest(q_i,q')$
    end if
  end for
end for

A heuristic partitioning strategy guided by a transition (q,q') is used: the region is partitioned in two subregions using a hyperplane of the beacon $\mathscr{B}_{qq'}$. Notice that beacons will always be polytopes, as $Pre(\mathscr{P}_{q'})$ is a polytope for linear dynamics, $\mathbb{U}$ is a polytope and the intersection of two polytopes is a polytope. The hyperplane which maximizes the radius of the Chebyshev ball that can fit in any of the resulting regions is chosen as the partitioning criterion. Choosing a hyperplane of the beacon ensures that only one of the resulting states can have a transition to q'. Even if a controller that enables the transition to q' does not exist for this state, after further partitioning the beacon becomes a state itself and the transition is enabled for it. The employed maximal radius criterion is likely to result in a less-complex partition, as opposed to iteratively computing one-step controllable sets to $\mathscr{B}_{qq'}$, and it is applicable to high dimensional state-spaces.

Let $\mathscr{A}^{D_i} = (Q^{D_i}, \rightarrow^{D_i}, \Gamma^{D_i}, \tau^{D_i}, Q_0^{D_i}, F^{D_i})$ denote the dual automaton after refinement iteration i, and let $\mathscr{A}^{D_0}$ denote the initial dual automaton. For a dual automaton $\mathscr{A}^{D_i}$, the set $\mathbb{X}_0^i \subseteq \mathbb{X}$ denotes the union of the regions corresponding to start states of automaton $\mathscr{A}^{D_i}$ with finite path costs, i.e.,

$$\mathbb{X}_0^i := \bigcup_{q \in \{q' \in Q_0^{D_i} \mid J^P(q') < \infty\}} \mathscr{P}_q \subseteq \mathbb{X}. \tag{7}$$

Example 4.4 Consider system (2) with $A = I_2$, $B = I_2$, $\mathbb{U} = \{u \in \mathbb{R}^2 \mid 0 \leq u_1 \leq 0.2, -0.1 \leq u_2 \leq 0.2\}$ and specification from[1] Ex. 4.3. $\mathscr{A}^{D_0}$ has two states $\{q_1,q_2\}$; both are initial states and q_2 is a final state. Since $J((q_1,q_2)) = \infty$, initially only q_2 has finite cost and q_1 is a candidate state for partitioning. Using a hyperplane of $\mathscr{B}_{q_1q_2}$ generates the state regions and the automaton shown in Fig. 4c and Fig. 4d. As $Post(\mathscr{P}_{q_3}) \cap \mathscr{P}_{q_2} = \emptyset$, the transition (q_3,q_2) is removed. In the next iteration q_1 is partitioned using $\mathscr{B}_{q_1q_2}$ and the algorithm terminates after this iteration, since there exists a finite cost automaton path from all states to the final state and the candidate set is empty. The control synthesis tools of Sec. 5 were used in this example to check the costs of the transitions.

Proposition 4.5 *Assume $\mathbb{X}_0^0$ is non-empty. Given an arbitrary iteration $i \geq 1$ of Alg. 2, the set $\mathbb{X}_0^i$ as defined in Eqn.* (7) *has the following properties:*

(i) *There exists a sequence of admissible control actions such that every closed-loop trajectory of system* (2) *originating in $\mathbb{X}_0^i$ satisfies formula Φ, and*

(ii) $\mathbb{X}_0^{i-1} \subseteq \mathbb{X}_0^i$.

PROOF. *(i)* A finite path cost for a state $q_0 \in Q_0^{D_i}$ implies that there exists an automaton run $q_0q_1\dots q_d$ with $J((q_j,q_{j+1})) < \infty$ for all $j = 0,\dots,d-1$ and $J^P(q_0) = \sum_{j=0}^{d-1} J((q_j,q_{j+1}))$. As a transition cost is assigned according to the existence of the controller that enables the transition, there exists a control sequence that ensures that every closed-loop trajectory originating in $\mathscr{P}_{q_0}$ reaches $\mathscr{P}_{q_d}$ by following the automaton path. Considering that removing states and transitions only reduces the language of the automaton, by Prop. 4.2 it follows that $\mathscr{L}_{\mathscr{A}^{D_0}} \subseteq \mathscr{L}_\Phi$. Since the proposed partitioning procedure preserves the language, we have $\mathscr{L}_{\mathscr{A}^{D_i}} \subseteq \mathscr{L}_{\mathscr{A}^{D_{i-1}}}$. Consequently, $\mathscr{L}_{\mathscr{A}^{D_i}} \subseteq \mathscr{L}_\Phi$ and the resulting trajectories satisfy the formula.

(ii) For any $x \in \mathbb{X}_0^{i-1}$, there exists an accepting automaton run $r_D = q_0q_1\dots q_d$ with $x \in \mathscr{P}_{q_0}$ and $J^P(q_0) = \sum_{j=0}^{d-1} J((q_j,q_{j+1})) < \infty$. Let q_s be the state chosen for partitioning at iteration i. Then, $J^P((q_s)) = \infty$ and $q_s \neq q_j$ for all $j = 0,\dots,d$ as $J^P(q_j) < \infty$ for all $j = 0,\dots,d$. As only the transitions adjacent to q_s are affected by partitioning, $q_j \in Q_i^D$ for all $j = 0,\dots,d$ and $(q_j,q_{j+1}) \in \rightarrow^{D_i}$ for all $j = 0,\dots,d-1$. Therefore, $x \in \mathscr{P}_{q_0}$, $r_D = q_0q_1\dots q_d$ is an accepting run of A^{D_i} with finite cost and thus, $x \in \mathbb{X}_0^i$. Observing that $x \in \mathbb{X}_0^{i-1}$ was chosen arbitrary completes the proof. □

[1] Note that the automata in Fig. 2 and in Fig. 6a represent $\mathscr{A}^{D_0}$. For simplicity the final state labeled by T is not shown in Fig. 6a.

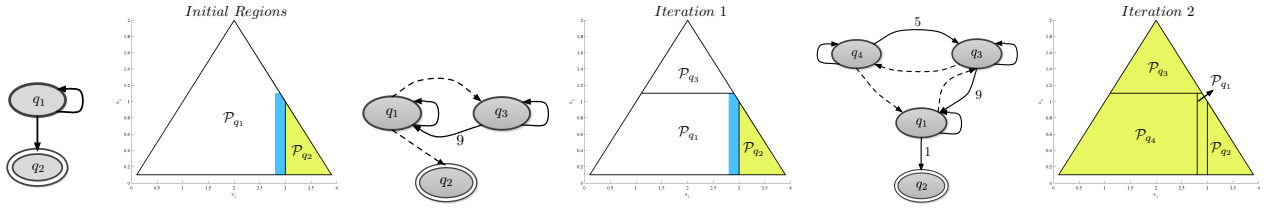


Figure 4: Automata and their corresponding polytopic state-space partitions for the iterations of Ex. 4.4. The polytopes are shown with black borders and $\mathbb{X}_0^i$ is shown in yellow. The beacon of transition (q_1,q_2) is shown in blue. A transition is shown with dashed line if a controller that enables the transition was not found, otherwise the transition is marked with a time bound.

The automaton refinement algorithms presented in this section generate a finite set of polytope-to-polytope control problems. Several tractable approaches for solving these problems are proposed in the next section.

5. POLYTOPE-TO-POLYTOPE CONTROL

Enabling a transition (q,q') requires an admissible control law that solves the $\mathscr{P}_q$-to-$\mathscr{P}_{q'}$ control problem. By the definition of $\mathscr{B}_{qq'}$, this problem can be decomposed in two subproblems. The first problem concerns the computation of a control law which generates a closed-loop trajectory, for all $x \in \mathscr{B}_{qq'}$, that reaches $\mathscr{P}_{q'}$ in one discrete-time instant. The second problem concerns the construction of a control law which generates a closed-loop trajectory, for all $x \in \mathscr{P}_q$, that reaches $\mathscr{B}_{qq'}$ in a finite number of discrete-time instants. These synthesis problems are formally stated next.

Problem 5.1 Let $\mathscr{B},\mathscr{P} \in \mathscr{P}(\mathbb{R}^n)$ with $\mathscr{B} \subseteq Pre(\mathscr{P})$ and consider system (2). Construct a state-feedback control law $g : \mathbb{R}^n \to \mathbb{R}^m$ such that

$$Ax+Bg(x) \in \mathscr{P},\ g(x) \in \mathbb{U}, \qquad \forall x \in \mathscr{B}.$$

Problem 5.2 Let $N \in \mathbb{Z}_{\geq 1}$, $\mathscr{B},\mathscr{P} \in \mathscr{P}(\mathbb{R}^n)$ with $\mathscr{B} \subseteq \mathscr{P}$ and consider system (2). Construct a state-feedback control law $g : \mathbb{R}^n \to \mathbb{R}^m$ such that for all $x \in \mathscr{P}$ it holds that

$$\begin{aligned} x_0 &:= x,\\ x_{k+1} &= Ax_k+Bg(x_k), \quad \forall k=0,\dots,N-1,\\ x_k &\in \mathscr{P},\ g(x_k) \in \mathbb{U}, \quad \forall k=0,\dots,N-1,\\ x_N &\in \mathscr{B}. \end{aligned}$$

Notice that while Prob. 5.1 is always feasible since $\mathscr{B} \subseteq Pre(\mathscr{P})$, Prob. 5.2 needs not be feasible for any set $\mathscr{P}$ and corresponding beacon $\mathscr{B}$. In what follows, sufficient conditions for feasibility of Prob. 5.2 will be indicated.

Firstly, let us present a vertex interpolation-based solution to Prob. 5.1. Let $\{v^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{B})}$ denote the set of vertices of $\mathscr{B}$ and let $\{u^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{B})}$ denote a corresponding set of control actions. Consider the following set of linear inequalities in the variables $\{u^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{B})}$:

$$\begin{aligned} H_{\mathscr{P}}(Av^i+Bu^i) &\leq h_{\mathscr{P}},\\ H_{\mathbb{U}}u^i &\leq h_{\mathbb{U}}. \end{aligned} \tag{8}$$

A solution to (8) can be obtained *off-line* by solving a feasibility LP. It is trivial to deduce that the control law

$$g(x) := \sum_{i=1}^{\#\mathscr{V}(\mathscr{B})} \lambda_i u^i, \tag{9}$$

where $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, are such that $x = \sum_{i=1}^{\#\mathscr{V}(\mathscr{B})} \lambda_i v^i$, solves Prob. 5.1. The evaluation of the control law (9) requires *on-line* calculation of the coefficients $\{\lambda_i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{B})}$, which amounts to solving a system of linear equations and can also be formulated as a feasibility LP.

Alternatively, an explicit piecewise affine (PWA) form of $g(\cdot)$ can be obtained by a simplicial partition of $\mathscr{B}$. Then, the evaluation of g requires solving *on-line* a point location problem, which consists of checking a finite number of linear inequalities. Although efficient ways to solve point location problems exist, depending on the complexity of the partition (number of simplices), the point location problem may be more computationally expensive than calculating the coefficients $\{\lambda_i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{B})}$ *on-line*. Yet another explicit PWA solution to Prob. 5.1 can be obtaining via direct synthesis of a PWA control law defined over an arbitrary polytopic partition of $\mathscr{B}$, which can still be formulated as a LP. While this approach may lead to a less complex point location problem, however, feasibility is not necessarily guaranteed for an arbitrary partition.

Next, two approaches are proposed to solve Prob. 5.2, i.e., vertex interpolation and polyhedral LFs.

Vertex interpolation Let $\{v^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})}$ be the vertices of $\mathscr{P}$ and let $\{\mathbf{u}^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})}$ denote a corresponding set of finite sequences of control actions, where $\mathbf{u}^i := \{u_k^i\}_{k=0,\dots,N-1}$ for all $i \in \mathbb{Z}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})}$ and $N \in \mathbb{Z}_{\geq 1}$. For each $v^i \in \mathscr{V}(\mathscr{P})$ define the following set of linear equality and inequality constraints in the variables $\{\mathbf{u}^i\}_{i\in\mathbb{Z}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})}}$:

$$\begin{aligned} x_0^i &:= v^i,\\ x_{k+1}^i &= Ax_k^i+Bu_k^i, \quad \forall k=0,\dots,N-1,\\ H_{\mathscr{P}}x_k^i &\leq h_{\mathscr{P}},\ H_{\mathbb{U}}u_k^i \leq h_{\mathbb{U}}, \quad \forall k=0,\dots,N-1,\\ H_{\mathscr{B}}x_N^i &\leq h_{\mathscr{B}}. \end{aligned} \tag{10}$$

A solution to the set of problems (10) can be searched for *off-line* by solving repeatedly a corresponding set of feasibility LPs starting with $N=1$, for all $i=1,\dots,\#\mathscr{V}(\mathscr{P})$, and increasing N until a feasible solution is obtained for all LPs and the same value of N. Let $N^* \geq 1$ denote the minimal N for which a feasible solution was found. Then, it is straightforward to establish that for any $x \in \mathscr{P}$, the control law

$$g(x_k) := \sum_{i=1}^{\#\mathscr{V}(\mathscr{P})} \lambda_i u_k^i, \quad k=0,\dots,N^*-1, \tag{11}$$

where $x_0 = x$ and $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, are such that $x = \sum_{i=1}^{\#\mathscr{V}(\mathscr{P})} \lambda_i v^i$, solves Prob. 5.2 and yields closed-loop trajectories that reach $\mathscr{B}$ in at most N^* discrete-time instants.

Evaluation of the control law g of (11) at time $k=0$ requires *on-line* calculation of the coefficients $\{\lambda_i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})}$, which is a LP, while at every $k=1,\dots,N^*-1$ the analytic expression of g is

implemented. However, a faster convergence to $\mathscr{B}$ can be obtained by taking $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, such that $x = \sum_{i=1}^{\#\mathscr{V}(\mathscr{P})} \lambda_i x_{j^*}^i$, where

$$j^* := \arg\max\{ j \in \{0,\dots,N^*\} \mid x \in \mathrm{Co}(\{x_j^i\}_{i=1,\dots,\#\mathscr{V}(\mathscr{P})})\}.$$

Then, the resulting closed-loop trajectories will reach $\mathscr{B}$ in at most $N^* - j^*$ discrete-time instants.

Similarly as in the case of the control law (9), simplicial decompositions of $\mathscr{P}$ can be employed to obtain an explicit PWA form of the control law $g(x_k)$, $k = 0,\dots,N^*-1$, both for its standard and faster variants presented above.

Remark 5.3 In general, existence of a finite, common N such that all LPs (10) are feasible is not guaranteed. If a certain upper bound on N is reached, the *off-line* synthesis procedure is stopped and Alg. 3 is employed to further partition the set $\mathscr{P}$. In the "worst" case, the partitioning converges to the maximal controllable subset of $\mathscr{P}$ with respect to $\mathscr{B}$, which ultimately recovers the "one-step controllable sets" partition of the state-space. However, if a solution is found for a finite N, there is no need to further partition $\mathscr{P}$, which can result in a significant complexity reduction, as it is illustrated for the case studies presented in Sec. 7.

Sufficient conditions for feasibility of the LPs (10) can be obtained as follows. Consider the set

$$\mathscr{E}_{\mathscr{B}} := \{x^s \in \mathrm{int}(\mathscr{B}) \mid \exists u^s \in \mathbb{U} : x^s = Ax^s + Bu^s\}.$$

If $\mathscr{E}_{\mathscr{B}} \neq \emptyset$ and the Minkowski function of the polytope $\mathscr{P}$ is a local control Lyapunov function [24] for system (2), there always exists a $N_i \geq 1$ such that the LPs (10) are feasible for all $i = 1,\dots,\#\mathscr{V}(\mathscr{P})$. Then interpolation becomes feasible as control sequences of equal length can be obtained via augmentation with a suitable control action $u^{s,i}$, which corresponds to some $x^{s,i} \in \mathscr{E}_{\mathscr{B}}$. Notice that the same assumptions were employed in [19], where only obstacle avoidance specifications were considered and polyhedral LFs were employed.

In this respect, the proposed vertex interpolation solution for solving Prob. 5.2 can be regarded as a relaxation of standard interpolation synthesis methods, where existence of a closed-loop equilibrium is assumed.

Polyhedral LFs However, if $\mathscr{E}_{\mathscr{B}} \neq \emptyset$, a simpler explicit PWA solution to Prob. 5.2 can be obtained via polyhedral LFs, see, e.g., [19, 25], as follows. Let $\mathscr{M}(x) := \max_{i=1,\dots,w} W_{i\bullet}(x - x^s)$, where $w \geq n+1$ is the number of lines of the matrix $H_{\mathscr{P}}$ and $x^s \in \mathscr{E}_{\mathscr{B}}$, denote the Minkowski function of the polytope $\mathscr{P}$. Next, consider the conic polytopic partition $\{\mathscr{C}_i\}_{i=1,\dots,w}$ of $\mathscr{P}$ induced by x^s, which is constructed as follows:

$$\mathscr{C}_i := \{x \in \mathscr{P} | (W_{i\bullet} - W_{j\bullet})(x - x^s) \geq 0,\ j = 1,\dots,w\} \cup \{x^s\}.$$

Notice that $\cup_{i=0,\dots,w} \mathscr{C}_i = \mathscr{P}$ and $\mathrm{int}(\mathscr{C}_i) \cap \mathrm{int}(\mathscr{C}_j) = \emptyset$ for all $i \neq j$. Let $\rho \in \mathbb{R}$ with $0 \leq \rho < 1$ denote a desired convergence rate. Consider the PWA control law

$$g(x) := K_i x + a_i \quad \text{if} \quad x \in \mathscr{C}_i \tag{12}$$

and the following feasibility LP in the variables $\{K_i, a_i\}_{i=1,\dots,w}$, to be solved *off-line*:

$$\begin{aligned}
&\rho W_{i\bullet}(x - x^s) - W_{j\bullet}(Ax + Bg(x) - x^s) \geq 0, \\
&\qquad\qquad \forall x \in \mathscr{V}(\mathscr{C}_i), \forall j = 1,\dots,w, \\
&K_i x + a_i \in \mathbb{U}, \quad \forall i = 1,\dots,w, \\
&(A + K_i)x^s + a_i = x^s, \quad \forall i = 1,\dots,w.
\end{aligned} \tag{13}$$

Notice that ρ can be minimized to obtain an optimal convergence rate and a different ρ_i can be assigned to each cone $\mathscr{C}_i$, while (13) remains a LP.

Proposition 5.4 *Suppose that the LP* (13) *is feasible. Then the function $\mathscr{M}$ is a Lyapunov function and $\mathscr{P}$ is a ρ-contractive set for system* (2) *in closed-loop with the PWA control law* (12)*, with respect to the equilibrium $x^s \in \mathrm{int}(\mathscr{B})$.*

The proof of Prop. 5.4 is a straightforward application of Thm. III.6 from [25] and it is omitted for brevity.

Letting $k^* := \arg\min\{k \geq 1 \mid \rho^k \mathscr{P} \subseteq \mathscr{B}\}$, one obtains that all trajectories of system (2) in closed-loop with (12) that start in $\mathscr{P}$ reach $\mathscr{B}$ in at most k^* discrete-time instants. Thus, the PWA control law (12) solves Prob. 5.2. The *on-line* evaluation of (12) reduces to a point location problem that can be solved in logarithmic time due to the specific conic partition.

6. COMPLETE CONTROL STRATEGY

The proposed control strategy that solves Prob. 3.1 is composed of a finite state generator automaton $\mathscr{A}^C$ and a map M from transitions of $\mathscr{A}^C$ to state feedback controllers. The automaton $\mathscr{A}^C = (Q^C, \to^C, \Gamma^C, \tau^C, Q_0^C, F^C)$ is constructed from the dual automaton $\mathscr{A}^{D_R} = (Q^{D_R}, \to^{D_R}, \Gamma^{D_R}, \tau^{D_R}, Q_0^{D_R}, F^{D_R})$ and it results from Alg. 2 as follows:

$$\begin{aligned}
Q^C &= \{q \in Q^{D_R} \mid J^P(q) < \infty\}, \\
\to^C &= \{(q,q') \mid J((q,q')) < \infty, (q,q') \in \to^{D_R}\}, \\
\Gamma^C &= \Gamma^{D_R}, \\
\tau^C &: Q^C \to \Gamma^C, \quad \tau^C(q) = \{p \mid [p] \cap \mathscr{P}_q \neq \emptyset\}, \\
Q_0^C &= Q_0^D \cap Q^C, \\
F^C &= F^{D_R}.
\end{aligned} \tag{14}$$

The state feedback controllers assigned by M are constructed as described in Sec. 5. Existence of these controllers are guaranteed, since $\mathscr{A}^C$ has only finite cost transitions.

Given a state $x_0 \in \mathscr{P}_{q_0}$ of system (2) for some $q_0 \in Q_0^C$, there exists an accepting run $r_C = q_0 q_1 \dots q_d$ of $\mathscr{A}^C$. The run corresponds to a control sequence $M_{r_C} = M((q_0,q_1)),\dots,M((q_{d-1},q_d))$. Starting from $x_0 \in \mathscr{P}_{q_0}$, the state feedback controller $M((q_0,q_1))$ is applied to system (2) until the trajectory reaches $\mathscr{P}_{q_1}$. Then, the applied feedback controller switches to $M((q_1,q_2))$. This process continues until the trajectory reaches $\mathscr{P}_{q_d}$ while $M((q_{d-1},q_d))$ is applied.

The union of the regions corresponding to the initial states of automaton $\mathscr{A}^C$ defines the set of initial system states $\mathbb{X}_0$, such that closed-loop trajectories originating in $\mathbb{X}_0$ satisfy formula Φ:

$$\mathbb{X}_0 = \bigcup_{q \in Q_0^C} \mathscr{P}_q. \tag{15}$$

For a given accepting run $r_C = q_0 q_1 \dots q_d$ of $\mathscr{A}^C$, the time required to satisfy the specification for trajectories originating in $\mathscr{P}_{q_0}$ is upper bounded by $\sum_{i=0}^{d-1} J((q_i,q_{i+1}))$ when $M_{r_C} = M((q_0,q_1)),\dots,M((q_{d-1},q_d))$ is applied. If the control sequences are chosen according to shortest paths for each $q_0 \in Q_0^C$, the time required to satisfy the specification starting from any state $x_0 \in \mathbb{X}_0$ of system (2) is upper bounded by $\max_{q_0 \in Q_0^C} J^P(q_0)$. Moreover, the control sequences can also be chosen to minimize the number of controller switches. In this case, the number of maximum controller switches for the trajectories originating in $\mathbb{X}_0$ is bounded by $\max_{q_0 \in Q_0^C} J^L(q_0)$, where $J^L(q_0)$ is the minimal length of an accepting run of $\mathscr{A}^C$ starting from q_0.

The following theorem states that when the refinement algorithm terminates, the proposed solution to Prob. 3.1 is correct and complete.

Theorem 6.1 *Suppose Alg. 2 terminates, then any closed-loop trajectory that originates in $\mathbb{X}_0$ satisfies the formula Φ and any trajectory of system* (2) *that produces a word $w \in \mathscr{L}_\Phi$ originates in $\mathbb{X}_0$.*

PROOF. The proof that all the trajectories of the closed loop system satisfy the formula follows immediately from Prop. 4.5 since $\mathbb{X}_0 = \mathbb{X}_0^{D_R}$.

To show that any satisfying trajectory originates in $\mathbb{X}_0$, assume by contradiction that there exist $x_0 \notin \mathbb{X}_0$ such that $x_0 x_1 \dots x_d$ is a satisfying trajectory of system (2), i.e., $P_0 P_1 \dots P_d \in \mathscr{L}_\Phi$. Then by Prop. 4.2, there exists an accepting run $r_D = q_0 q_1 \dots q_d$ of the initial dual automaton $\mathscr{A}^{D_0}$ such that $x_k \in \mathscr{P}_{q_k}, \forall k = 0, \dots, d$. The run r_D induces a unique refined dual automaton run $r_{D^R} = q'_0 q'_1 \dots q'_d$ where q'_k and q_k coincide or q'_k is obtained from q_k through partitioning and $x_k \in \mathscr{P}_{q'_k} \subseteq \mathscr{P}_{q_k}$ for all $k = 0, \dots, d$.

Let $r'_{D^R} = q_{s_0} q_{s_1} \dots q_{s_{d'}}$ be obtained by eliminating consecutive duplicates in r_{D^R}. Then, for each $i = 0, \dots, d'-1$, $s_i < s_{i+1}$ and $x_k \in \mathscr{P}_{q_{s_i}}$ for all $k = s_i, s_i+1, \dots, s_{i+1}-1$. Then, $x_0 \notin \mathbb{X}_0$ indicates that $J^P(q_{s_0}) = \infty$. Hence, either $Post(\mathscr{P}_{q_{s_i}}) \cap \mathscr{P}_{q_{s_{i+1}}} = \emptyset$ or $J((q_{s_i}, q_{s_{i+1}})) = \infty$ for some $i = 0, \dots, d'$. Let s_i be the maximal index where $Post(\mathscr{P}_{q_{s_i}}) \cap \mathscr{P}_{q_{s_{i+1}}} = \emptyset$ or $J((q_{s_i}, q_{s_{i+1}})) = \infty$. Therefore, $J((q_{s_k}, q_{s_{k+1}})) < \infty$, $\forall k = i+1, \dots, d'-1$ and $J^P(q_{s_k}) < \infty$, $\forall k = i+1, \dots, d'$. As Alg. 2 terminates, it holds that

$$Post(\mathscr{P}_q) \bigcap \left\{ \bigcup_{J^P(q')<\infty, (q,q') \in \to^{D^R}} \mathscr{P}_{q'} \right\} = \emptyset \qquad (16)$$

for all q with $J^P(q) = \infty$. Consequently, $Post(\mathscr{P}_{q_{s_i}}) \cap \mathscr{P}_{q_{s_{i+1}}} = \emptyset$. As $x_{s_{i+1}-1} \in \mathscr{P}_{q_{s_i}}$ and $x_{s_{i+1}} \in \mathscr{P}_{q_{s_{i+1}}}$, there is no control $u \in \mathbb{U}$ that satisfies $x_{s_{i+1}} = A x_{s_{i+1}-1} + Bu$. Therefore $x_0 x_1 \dots x_d$ is not a trajectory of system (2) and thus, we reached a contradiction. □

Remark 6.2 As shown in Prop. 4.5, Alg. 2 establishes a set iteration which produces a monotonically increasing, with respect to set inclusion, sequence of sets described by unions of polytopes. Thm. 6.1 states that when this iteration converges in finite time then the maximal set of satisfying states has been obtained. This is possible whenever the maximal set is a polytope or a union of polytopes. As this is not necessarily the case for any specification, in practice, to guarantee finite time termination, an artificial stopping criterion can be used, such as, e.g., the size of the region of satisfying states of system (2).

Remark 6.3 The complexity of the proposed solution can be analyzed in two aspects: off-line and on-line parts. The complexity of the off-line part essentially depends on the number of iterations required to reach the stopping criterion of Alg. 2. At each iteration, Alg. 2 involves shortest path computation, basic polyhedral operations and linear programming. The on-line part deals with the generation of the control input for system (2) and involves linear programming.

7. IMPLEMENTATION AND CASE STUDIES

The proposed computational framework was implemented as a Matlab software package, which is freely downloadable from `hyness.bu.edu/software`. The toolbox takes as input a scLTL formula over a set of linear predicates, the matrices of a discrete-time linear system, and the control constraints set, and produces a solution to Prob. 3.1 in the form of a set of initial states and a state-feedback control strategy. The tool, which uses *scheck2* [22] for the construction of the FSA and MPT [26] for polyhedral operations, also allows for displaying the set of initial states and simulating the trajectories of the closed-loop system for 2D or 3D state-spaces.

7.1 Case Study 1 : Double Integrator

Obstacle avoidance for double integrators is a particularly challenging problem [18]. The discrete-time double integrator dynamics with sampling time of 1 second are of the form given in Eqn. (2), where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}. \qquad (17)$$

We assume that the control constraint set is given by $\mathbb{U} = \{u \mid -2 \le u \le 2\}$. The control specification is to visit region $\mathbb{A}$ or region $\mathbb{B}$, and then the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, and staying inside a safe region given by $\mathbb{X} = \{x \mid -10 \le x_1 \le 1.85, -10 \le x_2 \le 2\}$. The sets $\mathbb{X}, \mathbb{U}$ and the obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$ are the same as the ones used in [18]. All these polytopic regions, together with the linear predicates used in their definitions, are shown in Fig. 5 (a). Using these predicates, the specification can be written as the following scLTL formula:

$\Phi_2 = ((p_0 \wedge p_1 \wedge p_2 \wedge p_3 \wedge \neg(p_4 \wedge p_5) \wedge \neg(\neg p_5 \wedge \neg p_6 \wedge p_7)) \mathscr{U} (\neg p_8 \wedge p_9 \wedge \neg p_{10} \wedge p_{11})) \wedge (\neg(\neg p_8 \wedge p_9 \wedge \neg p_{10} \wedge p_{11}) \mathscr{U} ((p_5 \wedge \neg p_{12} \wedge \neg p_{13}) \vee (\neg p_5 \wedge \neg p_7 \wedge p_{14} \wedge p_{15})))$.

The FSA that accepts $\mathscr{L}_{\Phi_2}$ has 3 states and 6 transitions. The DNF simplification deletes 2425 conjunctive clauses with empty state regions. The initial dual automaton has 72 states and 2452 transitions; 3 of the states and 1921 of the transitions are removed via the pruning algorithm. After 183 iterations of the refinement algorithm, 228 of the dual automaton states have finite cost. The maximal set of initial states and a sample of satisfying trajectories of the closed loop system are shown in Fig. 6b. Every trajectory originating in $\mathbb{X}_0$ satisfies the specification within 20 discrete-time instants. The polytope-to-polytope controllers are synthesized using vertex interpolation. The computation took 10 minutes on a iMac with a Intel Core i5 processor at 2.8GHz with 8GB of memory.

As discussed in the paper, the upper time bound is affected by the choice of candidate polytopes for partitioning. In this example, a transition is selected from the candidate set according to the cost of the target state as described in Alg. 2. Our experiments showed that choosing the state with the highest Chebychev ball radius resulted in a faster coverage (117 iterations). However, it also produced a higher time bound of 28 steps.

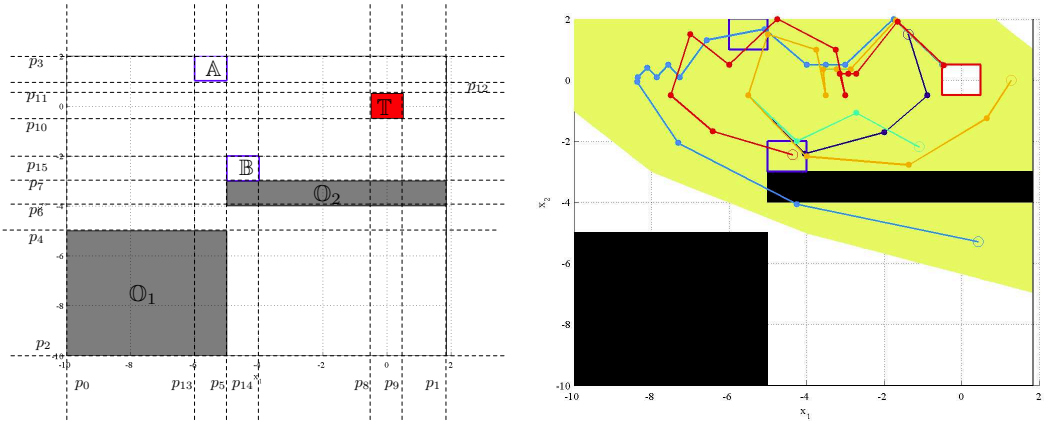
For the double integrator dynamics (17), the control strategy developed in this paper was also tested for a classical control specification, i.e., computation of the maximal constrained control invariant set within $\mathbb{X}$. The method converged to the actual maximal set for the dynamics (17) and the given sets $\mathbb{X}$ and $\mathbb{U}$, which is an indication of the non-conservatism of the vertex interpolation method that solves Prob. 5.2.

7.2 Case Study 2 : Triple Integrator

Consider a triple integrator with sampling time of 1 second, whose dynamics are described by Eqn. (2) with

$$A = \begin{bmatrix} 1 & 1 & 0.5 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.167 \\ 0.5 \\ 1 \end{bmatrix}, \qquad (18)$$

and $\mathbb{U} = \{u \in \mathbb{R} \mid -2 \le u \le 2\}$. The specification is to reach a target region $\mathbb{T}$, while always staying inside a safe set $\mathbb{X}$ and avoiding



(a) (b)

Figure 5: Case study 1: (a) The regions and the corresponding linear predicates. The predicates are shown in the half planes where they are satisfied; (b) The set of satisfying initial states $\mathbb{X}_0$ (yellow region) and some trajectories of the closed loop system (the initial states are marked by circles).

obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, where

$$\mathbb{T} = \{x \mid -0.5 \le x_i \le 0.5, i = 1,2,3\}$$
$$\mathbb{X} = \{x \mid -4 \le x_i \le 4, i = 1,2,3\}$$
$$\mathbb{O}_1 = \{x \mid 0.5 \le x_1 \le 4, -4 \le x_2 \le 4, -4 \le x_3 \le 0.5,\}$$
$$\mathbb{O}_2 = \{x \mid -4 \le x_1 \le -0.5, -4 \le x_2 \le 4, 2 \le x_3 \le 4\}.$$

These regions, which are all boxes (i.e., hyper-rectangular polytopes), are shown in Fig. 6 (b). Each box is represented using six predicates, one for each facet, where $[c_i^\top, c_{i+1}^\top, c_{i+2}^\top] = -I_3$, $i = 0,6$; $[c_i^\top, c_{i+1}^\top, c_{i+2}^\top] = I_3$, $i = 3,9$; $d_i = 4, i = 0,\ldots,5$; $d_i = 0.5 i = 6,\ldots,11$ and $c_{12} = -e_3$, $d_{12} = 2$. The specification can be formally stated as the following scLTL formula:

$\Phi_3 = (p_0 \wedge p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5 \wedge \neg(p_3 \wedge \neg p_9 \wedge p_1 \wedge p_4 \wedge p_2 \wedge p_{11}) \wedge \neg(p_0 \wedge \neg p_6 \wedge p_1 \wedge p_4 \wedge p_{12} \wedge p_5)) \mathscr{U} (p_6 \wedge p_7 \wedge p_8 \wedge p_9 \wedge p_{10} \wedge p_{11})$.

The FSA that accepts $\mathscr{L}_{\Phi_3}$ has 2 states and 3 transitions. The DNF simplification deletes 21 conjunctive clauses with empty state regions. The initial dual automaton has 16 states and 225 transitions, and 101 of the transitions are removed via the pruning algorithm. The refinement algorithm terminates after 4790 iterations and the refined dual automaton has 3612 states with finite cost. $\mathbb{X}_0$ and a sample of satisfying trajectories are shown in Fig. 6. Note that $\mathbb{X}_0$ covers 37% of the obstacle free safe region and any trajectory originating in $\mathbb{X}_0$ satisfies the specification in less than 10 steps. This computation took approximatively 5 hours using the same computer as in the Case Study 1.

In this experiment, the candidate states for partitioning are chosen according to the Chebychev ball radius. This example presents a worst case scenario for the developed framework, since most of the encountered controller synthesis problems of the type Prob. 5.2 were infeasible, and the states were partitioned until most of them became a beacon for a transition. Only 162 out of 6578 transition controllers were not one-step controllers.

8. CONCLUSIONS

This paper considered the problem of controlling discrete-time linear systems from specifications given as formulas of syntactically co-safe linear temporal logic over linear predicates in the state variables of the system. A systematic procedure was developed for the automatic computation of sets of initial states and feedback controllers such that all the resulting trajectories of the corresponding closed-loop system satisfy the given specifications. The developed procedure is based on the iterative construction and refinement of an automaton that enforces the satisfaction of the formula. Interpolation and polyhedral Lyapunov function based approaches were proposed to compute the polytope-to-polytope controllers for the transitions of the automaton. The algorithms developed in this paper were implemented as a software package that is available for download. Their application and effectiveness were demonstrated for two challenging case studies.

9. ACKNOWLEDGEMENTS

We would like to acknowledge the help of Dr. Amit Bhatia and Dr. Lydia E. Kavraki for sharing useful modifications of the scheck code. This work was partially supported at Boston University by the NSF under grants CNS-0834260 and CNS-1035588, by the ONR under grants 014-001-0303-5 and N00014-10-10952, and by the ARO under grant W911NF-09-1-0088.

The second author gratefully acknowledges the support of the Veni grant 10230 from the Dutch organizations NWO and STW.

10. REFERENCES

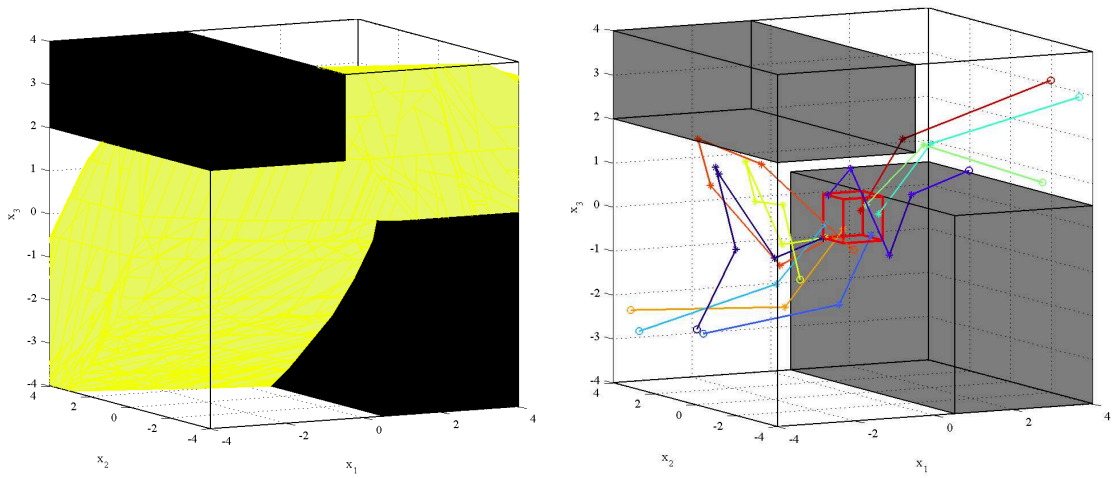
[1] E. M. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[2] A. Girard, "Synthesis using approximately bisimilar abstractions: state-feedback controllers for safety specifications," in *Hybrid Systems: Computation and Control*. ACM, 2010, pp. 111–120.

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

[4] G. E. Fainekos and A. Girard, "Hierarchical synthesis of hybrid controllers from temporal logic specifications," in *Hybrid Systems: Computation and Control*. Springer, 2007, pp. 203–216.

[5] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *IEEE Conf. on Decision and Control*, Shanghai, China, 2009, pp. 5997–6004.



(a) (b)

Figure 6: Case study 2: (a) The set of satisfying initial states is shown in yellow; (b) Some satisfying trajectories of the closed loop system (the initial states are marked by circles).

[6] P. Tabuada and G. Pappas, "Model checking LTL over controllable linear systems is decidable," ser. Lecture Notes in Computer Science, O. Maler and A. Pnueli, Eds. Springer-Verlag, 2003, vol. 2623.

[7] 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," in *Thirteen International Conference on Intelligent Systems for Molecular Biology*, 2005.

[8] M. Antoniotti, F. Park, A. Policriti, N. Ugel, and B. Mishra, "Foundations of a query and simulation system for the modeling of biochemical and biological processes," ser. Proceedings of the Pacific Symposium on Biocomputing, Lihue, Hawaii, 2003, pp. 116–127.

[9] G. Batt, C. Belta, and R. Weiss, "Temporal logic analysis of gene networks under parameter uncertainty," *IEEE Trans. on Circuits and Systems and IEEE Trans. on Automatic Control, joint special issue on Systems Biology*, vol. 53, pp. 215–229, 2008.

[10] H. K. Gazit, G. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *IEEE Conference on Robotics and Automation*, Rome, Italy, 2007.

[11] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *IEEE Conf. on Decision and Control*, Shanghai, China, 2009, pp. 2222 –2229.

[12] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot motion planning: A timed automata approach," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004, pp. 4417–4422.

[13] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[14] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Motion planning with hybrid dynamics and temporal goals," in *IEEE Conf. on Decision and Control*, Atlanta, GA, 2010, pp. 1108–1115.

[15] A. P. Sistla, "Safety, liveness and fairness in temporal logic," *Formal Aspects of Computing*, vol. 6, pp. 495–511, 1994.

[16] J. Tumova, B. Yordanov, C. Belta, I. Cerna, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *IEEE Conf. on Decision and Control*, Atlanta, GA, 2010, pp. 4230 –4235.

[17] S. V. Rakovic, F. Blanchini, E. Cruck, and M. Morari, "Robust obstacle avoidance for constrained linear discrete time systems: A set-theoretic approach," in *IEEE Conf. on Decision and Control*, 2007, pp. 188 –193.

[18] S. V. Rakovic and D. Mayne, "Robust Model Predictive Control for Obstacle Avoidance: Discrete Time Case," *Lecture Notes in Control and Information Sciences (LNCIS)*, vol. 358, pp. 617–627, Sep. 2007.

[19] F. Blanchini, S. Miani, F. Pellegrino, and B. van Arkel, "Enhancing controller performance for robot positioning in a constrained environment," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 5, pp. 1066 –1074, 2008.

[20] G. M. Ziegler, *Lectures on polytopes*. Springer, 2007.

[21] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, pp. 291–314, 2001.

[22] T. Latvala, "Efficient model checking of safety properties," in *In Model Checking Software. 10th International SPIN Workshop*. Springer, 2003, pp. 74–88.

[23] T. Cormen, *Introduction to Algorithms*. MIT press, 2001.

[24] V. Spinu, M. Lazar, and P. P. J. van den Bosch, "An explicit state-feedback solution to constrained stabilization of DC-DC power converters," in *IEEE Conference on Control Applications*, september 2011, pp. 520 –525.

[25] M. Lazar, "On infinity norms as Lyapunov functions: Alternative necessary and sufficient conditions," in *IEEE Conf. on Decision and Control*, 2010, pp. 5936 –5942.

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