Language-guided controller synthesis for linear systems

Language-Guided Controller Synthesis for Linear Systems

Ebru Aydin Gol, *Student Member, IEEE*, Mircea Lazar, *Member, IEEE*, and Calin Belta, *Senior Member, IEEE*

***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. 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 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. Linear programming based approaches are proposed to compute the polytope-to-polytope controllers that label the transitions of the automaton. Extensions to discrete-time piecewise affine systems and specifications given as formulas of full linear temporal logic are included. The algorithms developed in this paper were implemented as a software package that is available for download. Their application and effectiveness are demonstrated for several case studies.**

***Index Terms*—Constrained control, linear temporal logic (LTL).**

I. 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 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].

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 "$(\neg O \mathcal{U} T) \wedge (\neg T \mathcal{U} (R_1 \vee R_2))$" requires convergence to target region T through regions R_1 or R_2 while avoiding obstacle O. For simplicity of presentation, throughout the paper we focus on linear systems. We show, however, that the results can be easily extended to discrete-time piecewise affine systems. We also discuss the extension to specifications given as LTL formulas.

Central to our 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 automaton produces the language satisfying the specification, hence the name *language guided* for our approach. 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 linear programming 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 main contributions of this work are the following. 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]. Second, this paper provides two solutions based on linear programming for solving polytope-to-polytope control problems. The first solution is based on vertex interpolation and requires iteratively solving a finite

Manuscript received August 10, 2012; revised February 14, 2013; accepted November 26, 2013. Date of publication December 18, 2013; date of current version April 18, 2014. This work was supported in part by the National Science Foundation (NSF) under grants CNS-0834260 and CNS-1035588, by the ONR under grants 014-001-0303-5 and N00014-10-10952, and by the Veni grant 10230 from the Dutch organizations NWO and STW. Recommended by Associate Editor D. Hristu-Varsakelis.

E. A. Gol and C. Belta are with the Division of Systems Engineering, Boston University, Boston, MA 02215 USA (e-mail: ebru@bu.edu; cbelta@bu.edu).

M. Lazar is with the Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven 5600MB, The Netherlands (e-mail:m.lazar@tue.nl).

Digital Object Identifier 10.1109/TAC.2013.2295664

number of linear programs. The second solution is based on contractive sets. While more conservative, it only requires solving a single linear program. The existing approaches on solving polytope-to-polytope control problems for discrete-time systems are based on iterative computations of one-step controllable sets, e.g., [17]. For continuous-time systems the problem of controlling a linear system from one polytope to another is defined as a facet reachability problem [18], [19]. Similar to the proposed vertex interpolation method, to solve the facet reachability problem, first controls for the vertices are computed via linear programming, and then these controls are used to define a state feedback control law. While facet reachability is enforced by a flow constraint in the linear program [18], in the vertex interpolation method reachability of the polytope is enforced by direct constraints on the trajectories originating at the vertices.

This paper extends current results on obstacle avoidance [17], [20], [21]. 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. 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 proposed computational framework was implemented as a Matlab software package, which is freely downloadable from hyness.bu.edu/software together with the examples presented in the paper.

A preliminary version of this work appeared in [22]. Here, we expand this preliminary version in three main directions. First, we propose a novel and efficient method to solve the polytope-to-polytope control problem. Second, we show how the proposed language-guided approach can be extended to piecewise-affine (PWA) systems and specifications given as formulas of full LTL. Third, we provide extensive analysis of complexity.

The paper is organized as follows. We review some notions necessary throughout the paper in Section II before formulating the problem and outlining the approach in Section III. The iterative construction of the abstraction is presented in Section IV. The LP-based algorithms for solving polytope-to-polytope control problems are described in Section V. The main theorem is stated in Section VI. The extensions to PWA systems and full LTL specifications are discussed in Section VII, while illustrative examples are shown in Section VIII. Conclusions are summarized in Section IX.

II. Notation and Preliminaries

For a set $\mathcal{S}$, $\mathrm{int}(\mathcal{S})$, $\mathrm{Co}(\mathcal{S})$, $\#\mathcal{S}$, and $2^{\mathcal{S}}$ stand for its interior, convex hull, cardinality, and power set, respectively. For $\lambda \in \mathbb{R}$ and $\mathcal{S} \subset \mathbb{R}^n$, let $\lambda\mathcal{S} := \{\lambda x | x \in \mathcal{S}\}$. 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 two arbitrary sets $\mathcal{S}, \mathcal{P} \subset \mathbb{R}^n$, $\mathcal{S} \oplus \mathcal{P} = \{s+p \mid s \in \mathcal{S}, p \in \mathcal{P}\}$ denotes their Minkowski sum.

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 $\mathcal{V}(\mathcal{P})$ to denote the set of vertices of a polytope $\mathcal{P}$. Both the $\mathcal{V}$-representation $(\mathrm{Co}(\mathcal{V}(\mathcal{P})))$ and the $\mathcal{H}$-representation $(\{x \in \mathbb{R}^n \mid H_{\mathcal{P}}x \leq h_{\mathcal{P}}\}$, where matrix $H_{\mathcal{P}}$ and vector $h_{\mathcal{P}}$ have suitable dimensions) [23] of a polytope $\mathcal{P}$ will be used throughout the paper.

Definition II.1: [24] An scLTL formula over a set of atomic propositions P is inductively defined as follows:

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

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

An infinite word w over a finite set Σ is an infinite sequence $w = w_0 w_1 \ldots$, where $w_i \in \Sigma$ for all $i \in \mathbb{Z}_+$. Similarly, a finite word w over a finite set Σ is a finite sequence $w = w_0 \ldots w_d$, where $d \in \mathbb{Z}_+$ and $w_i \in \Sigma$ for all $i = 0, \ldots, d$. The semantics of scLTL formulas is defined over infinite words over 2^P as follows:

Definition II.2: The satisfaction of an 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) $w_i \models \Phi_1 \mathcal{U} \Phi_2$ 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) $w_i \models \diamondsuit\, \Phi$ if there exists $j \geq i$ such that $w_j \models \Phi$. A word w satisfies an 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[1]. We use $\mathcal{L}_\Phi$ to denote the set of all (finite) prefixes of all satisfying infinite words.

Definition II.3: A deterministic finite state automaton (FSA) is a tuple $\mathcal{A} = (Q, \Sigma, \rightarrow_{\mathcal{A}}, Q_0, F)$, where Q is a finite set of states, Σ is a set of symbols, $\rightarrow_{\mathcal{A}} \subseteq Q \times \Sigma \times Q$ is a deterministic transition relation, $Q_0 \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is a set of final states.

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

For any scLTL formula Φ over P, there exists a FSA $\mathcal{A}$ with input alphabet 2^P that accepts the prefixes of all the satisfying words, i.e., $\mathcal{L}_\Phi$[24]. There are algorithmic procedures and off-the-shelf tools, such as *scheck2*[25], for the construction of such an automaton.

Definition II.4: Given a FSA $\mathcal{A} = (Q, \Sigma, \rightarrow_{\mathcal{A}}, Q_0, F)$, its dual automaton is a tuple $\mathcal{A}^D = (Q^D, \rightarrow^D, \Sigma, \tau^D, Q_0^D, F^D)$ where $Q^D = \{(q, \sigma, q') \mid (q, \sigma, q') \in \rightarrow_{\mathcal{A}}\}$, $\rightarrow^D = \{((q, \sigma, q'), (q', \sigma', q'')) \mid (q, \sigma, q'), (q', \sigma, q'') \in \rightarrow_{\mathcal{A}}\}$, $\tau^D : Q^D \rightarrow \Sigma$, $\tau^D((q, \sigma, q')) = \sigma$, $Q_0^D = \{(q_0, \sigma, q) \mid q_0 \in Q_0\}$, and $F^D = \{(q, \sigma, q') \mid q' \in F\}$.

Informally, the states of the dual automaton $\mathcal{A}^D$ are the transitions of the automaton $\mathcal{A}$. $\rightarrow^D \subseteq Q^D \times Q^D$ is a transition relation and a transition is defined between two states of $\mathcal{A}^D$ if

[1]We abuse the terminology and say that a finite word satisfies a formula if it contains a satisfying finite prefix.

the corresponding transitions are connected by a state in $\mathcal{A}$. The set of output symbols of $\mathcal{A}^D$ is the same as the set of symbols of $\mathcal{A}$, i.e., Σ. τ^D is an output function. For a state of $\mathcal{A}^D$, τ^D produces the symbol that enables the transition in $\mathcal{A}$. The set of initial states Q_0^D of $\mathcal{A}^D$ is the set of all transitions that leave an initial state in $\mathcal{A}$. Similarly, the set of final states F^D of $\mathcal{A}^D$ is the set of transitions that end in a final state of $\mathcal{A}$.

An accepting run $r_{\mathcal{A}}^D$ of a dual automaton is a sequence of states $r_{\mathcal{A}}^D = q_0 \dots q_d$ such that $q_0 \in Q_0^D$, $q_d \in F^D$ and $(q_i, q_{i+1}) \in \rightarrow_{\mathcal{A}}^D$ for all $i = 0, \dots, d-1$. An accepting run $r_{\mathcal{A}}^D$ produces a word $w = w_0 \dots w_d$ over Σ such that $\tau(q_i) = w_i$, for all $i = 0, \dots, d$. The output language $\mathcal{L}_{\mathcal{A}^D}$ of a dual automaton $\mathcal{A}^D$ is the set of all words that are generated by accepting runs of $\mathcal{A}^D$. The construction of a dual automaton $\mathcal{A}^D$ from a FSA $\mathcal{A}$ guarantees that any word produced by $\mathcal{A}^D$ is accepted by $\mathcal{A}$:

Proposition II.5: The output language of the dual automaton $\mathcal{A}^D$ coincides with the language accepted by the automaton $\mathcal{A}$, i.e., $\mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{A}^D}$.

Proof: $\Rightarrow$: For every word $w = w_0 \dots w_d$ accepted by automaton $\mathcal{A}$, there exists a run $r_{\mathcal{A}} = q_0 \dots q_{d+1}$ such that $q_0 \in Q_0$, $q_{d+1} \in F$ and $(q_i, w_i, q_{i+1}) \in \rightarrow_A$ for all $i = 0, \dots, d$. The transition sequence of this run corresponds to a run r_D of the dual automaton which generates the word w since $(q_0, w_0, q_1) \in Q_0^D$, $(q_d, w_d, q_{d+1}) \in F^D$ and $((q_{i-1}, w_{i-1}, q_i), (q_i, w_i, q_{i+1})) \in \rightarrow^D$ for all $i = 1, \dots, d$. $\Leftarrow$: Similarly, a run $r_D = (q_0, w_0, q_1) \dots (q_d, w_d, q_{d+1})$ of $\mathcal{A}^D$ that produces the word $w = w_0 \dots w_d$ yields a run $r_{\mathcal{A}} = q_0 \dots q_{d+1}$ of $\mathcal{A}$ which accepts the word w. ∎

III. Problem Formulation

Consider a discrete-time linear control system of the form[2]

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

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$ describe the system dynamics, $\mathbb{X} \subset \mathbb{R}^n$ and $\mathbb{U} \subset \mathbb{R}^m$ are polyhedral sets, and $x_k \in \mathbb{X}$ and $u_k \in \mathbb{U}$ 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., $P_i = \{p_j \mid x_i \in [p_j]\}$. scLTL formulas over the set of predicates P can therefore be interpreted over such words (see Section II). A system trajectory satisfies an scLTL formula over P if the word produced by the trajectory satisfies the corresponding formula.

Given an scLTL formula Φ over P, we use $\overline{\mathbb{X}_0^\Phi}$ to denote the set of all initial states of system (2) for which there exist control sequences producing trajectories satisfying Φ.

Problem III.1: Given an scLTL formula Φ over a set of linear predicates P and a dynamical system as defined in (2), construct a set of initial states $\mathbb{X}_0^\Phi \subseteq \overline{\mathbb{X}_0^\Phi}$ (possibly $\overline{\mathbb{X}_0^\Phi}$) and a feedback control strategy such that all the closed-loop trajectories originating in $\mathbb{X}_0^\Phi$ satisfy Φ.

We propose a solution to the above problem by relating the control synthesis problem with the construction of a dual automaton (Def. II.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 from the automaton that accepts the prefixes of all words satisfying formula Φ. Its states will be refined until feasible polytope-to-polytope controllers are obtained. This approach reduces the controller synthesis part of Prob. III.1 to solving a finite number of polytope-to-polytope control problems.

The proposed solutions to polytope-to-polytope controller synthesis give 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.

IV. Automaton Generation and Refinement

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

A. FSA and Dual Automaton

All words that satisfy the specification formula Φ are accepted by a FSA $\mathcal{A} = (Q, 2^P, \rightarrow_{\mathcal{A}}, Q_0, F)$. The dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, 2^P, \tau^D, Q_0^D, F^D)$ is constructed by interchanging the states and the transitions of the automaton $\mathcal{A}$ (Def. II.4). As the transitions of $\mathcal{A}$ become states of $\mathcal{A}^D$, elements from 2^P label the states and define polyhedral sets within the state-space of system (2).

1) Automaton Representation: A FSA $\mathcal{A}$ with input alphabet 2^P that accepts the language of an scLTL formula Φ over P is constructed with the tool *scheck2* [25]. This tool labels each transition of the produced FSA with a disjunctive normal form (DNF) $C_1 \vee C_2 \vee \dots \vee C_d$, 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.

Given a DNF formula $D = C_1 \vee C_2 \vee \dots \vee C_d$, $\mathcal{P}_{C_i} := [p_{i_1}] \cap \dots \cap [p_{i_c}]$ denotes the set of states of system (2) that satisfy $C_i = p_{i_1} \wedge \dots \wedge p_{i_c}$ where $i_c \in \{0, \dots, l\}$, $i_j \in \{0, \dots, i_c\}$ for all $j \leq c$, and $\mathcal{P}_D := \cup_{i=1}^d \mathcal{P}_{C_i}$ 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 $C_1 \vee C_2 \vee \dots \vee C_d$ is simplified by applying the following rules:

1) *Empty set elimination*: C_i is eliminated if the corresponding region is empty, i.e., $\mathcal{P}_{C_i} = \emptyset$. The symbols that satisfy such clauses can not be generated by the system trajectories.
2) *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., $\mathcal{P}_{C_i} \subseteq \mathcal{P}_{C_j}$. The system states that satisfy C_i also satisfy C_j, which enables the same transition.

[2]We focus on discrete-time linear systems to keep the notation to a minimum. In Section VII, we show how the solution can be easily extended to accommodate discrete-time piecewise affine systems.

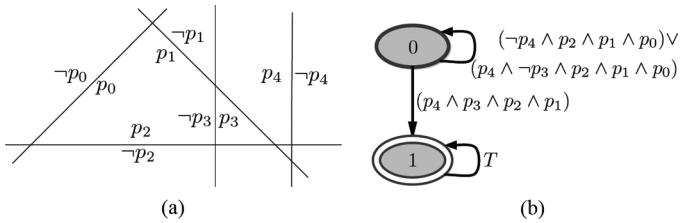


Fig. 1. (a) Half-spaces generated by the linear predicates in (4). (b) Compact representation of a FSA that accepts the language satisfying formula Φ_1 in (4). The initial states are filled with gray and the final state is marked with a double circle. T stands for the Boolean constant true.

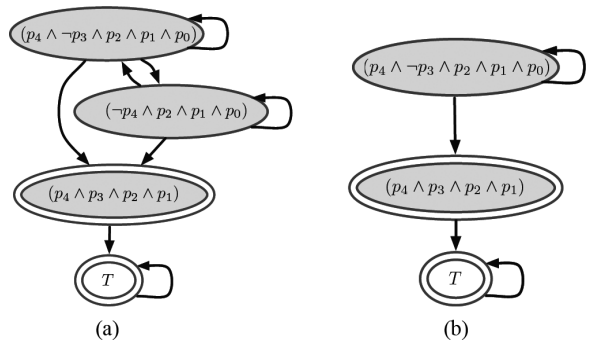


Fig. 2. Dual automata for the FSA from Fig. 1(b): (a) without Boolean simplification; (b) with Boolean simplification.

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.

In what follows, we use $\mathcal{P}_q \subseteq \mathbb{X}$ to denote the set of states of system (2) that satisfy the Boolean formula of a dual automaton state q.

Example IV.1: Consider the following scLTL formula:

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

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 and the compact representation of a FSA that accepts the language satisfying formula Φ_1 are shown in Fig. 1. 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.

The compact representations of dual automata constructed with and without simplifying the DNF formulas are shown in Fig. 2, 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. 1(b), since the set of states that satisfies this clause is empty. The transitions of dual automata are added with respect to graph structure of the FSA as defined in Def. II.4. For example, there is a transition from the state labeled with $(p_4 \wedge \neg p_3 \wedge p_2 \wedge p_1 \wedge p_0)$ to the states that correspond to transitions leaving the state labeled with "0".

An accepting run $r_D = q_0 \dots q_d$ of $\mathcal{A}^D$ defines a sequence of polyhedral sets $\mathcal{P}_{q_0} \dots \mathcal{P}_{q_d}$. Any trajectory $x_0 \dots x_d$ of system (2) with $x_i \in \mathcal{P}_{q_i}$, $i = 0, \dots, d$ satisfies the specification by Prop. II.5. We say that a transition (q, q') of $\mathcal{A}^D$ is *enabled* if there exists an admissible control law that achieves the transition for all $x \in \mathcal{P}_q$. 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 $\mathcal{P}_q$ reach $\mathcal{P}_{q'}$ in finite time and remain within $\mathcal{P}_q$ until they reach $\mathcal{P}_{q'}$. 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 $\mathcal{P}_q$ reaches $\mathcal{P}_{q'}$ at the next discrete-time instant. For every transition of $\mathcal{A}^D$, if a controller that enables the transition can be constructed, then every resulting closed-loop trajectory originating in $\cup_{q_0 \in Q_0^D} \mathcal{P}_{q_0}$ will satisfy the specifications by Prop. II.5. However, existence of such controllers is not guaranteed for all the states of system (2) within $\mathbb{X}$.

Prob. III.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 $\mathbb{X}$.

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. $Post(\mathcal{P})$ denotes the set of states that can be reached from $\mathcal{P}$ in one discrete-time instant under the dynamics (2) and $Post(\mathcal{P})$ is formally defined as

$$Post(\mathcal{P}) = \{Ax + Bu \mid (x, u) \in \mathcal{P} \times \mathbb{U}\}. \tag{5}$$

For a transition (q, q'), if $Post(\mathcal{P}_q) \cap \mathcal{P}_{q'} = \emptyset$, then this transition is considered *infeasible* , since there is no admissible controller that enables this transition. As $\mathcal{P}$ and $\mathbb{U}$ are polytopes, $Post(\mathcal{P})$ can be computed as follows:

$$Post(\mathcal{P}) = \mathrm{Co}(\{Ax + Bu \mid (x, u) \in \mathcal{V}(\mathcal{P}) \times \mathcal{V}(\mathbb{U})\}). \tag{6}$$

Alg. 1 summarizes the procedure. Once the infeasible transitions are removed as in line 2,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

Input: Dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, 2^P, \tau^D, Q_0^D, F^D)$.

Output: Pruned dual automaton $\mathcal{A}^{D_0} = (Q^{D_0}, \rightarrow^{D_0}, 2^P, \tau^D, Q_0^{D_0}, F^D)$

$Q^{D_0} := Q^D$

$\rightarrow^{D_0} := \rightarrow^D \ \{(q,q') | Post(\mathcal{P}_q) \cap \mathcal{P}_{q'} = \emptyset\}$

while $Q^D \neq \emptyset$ **do**

  for all $q \in Q^D$ **do**

    $Q^D := Q^D \setminus \{q\}$

    if $(q \notin F^D$ AND $\{q' \mid (q,q') \in \rightarrow^{D_0}\} = \emptyset)$ OR $(q \notin Q_0^D$ AND $\{q' \mid (q',q) \in \rightarrow^{D_0}\} = \emptyset)$ **then**

      $Q^{D_0} := Q^{D_0} \setminus \{q\}$

      $Q^D := Q^D \cup (\{q' \mid (q,q') \in \rightarrow^{D_0}\} \cup \{q' \mid (q',q) \in \rightarrow^{D_0}\})$

      $\rightarrow^{D_0} := \rightarrow^{D_0} \setminus (\{(q,q') | (q,q') \in \rightarrow^{D_0}\} \cup \{(q',q) | (q',q) \in \rightarrow^{D_0}\})$

    end if

  end for

end while

$Q_0^{D_0} = Q_0^D \cap Q^{D_0}$

B. Automaton Refinement

Alg. 1 guarantees that a non-empty polyhedral subset of a source polytope $\mathcal{P}_q$ is one-step controllable to the target polytope $\mathcal{P}_{q'}$ 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 $\mathcal{P}_q$ 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.

Algorithm 2 Refinement

Input: Dual automaton $\mathcal{A}^{D_0} = (Q^{D_0}, \rightarrow^{D_0}, 2^P, \tau^{D_0}, Q_0^{D_0}, F^{D_0})$.

Output: Refined dual automaton $\mathcal{A}^{D_R} = (Q^{D_R}, \rightarrow^{D_R}, 2^P, \tau^{D_R}, Q_0^{D_R}, F^{D_R})$

$J((q,q')) := FeasibilityTest(q,q')$, for each $(q,q') \in \rightarrow^{D_0}$

$J^P := ShortestPath(J, F^{D_0})$

$CandidateSet = \{(q_i,q_j) \mid (q_i,q_j) \in \rightarrow^{D_0}, J^P(q_i) = \infty, J^P(q_j) \neq \infty\}$

$k := 0$

while $CandidateSet \neq \emptyset$ **do**

  $(q_s,q_d) := \min_{J^P(q_j)}\{(q_i,q_j) \mid (q_i,q_j) \in CandidateSet\}$

  $\mathcal{A}^{D_{k+1}} := Partitioning(\mathcal{A}^{D_k}, q_s)$

  $J((q,q')) := FeasibilityTest(q,q')$, for each $(q,q') \in \rightarrow^{D_{k+1}}$

  $J^P := ShortestPath(J, F^{D_{k+1}})$

  $CandidateSet := \{(q_i,q_j) \mid (q_i,q_j) \in \rightarrow^{D_{k+1}}, J^P(q_i) = \infty, J^P(q_j) \neq \infty\}$

  $k := k+1$

end while

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

$$Pre(\mathcal{P}) \coloneqq \{x \in \mathbb{X} \mid \exists u \in \mathbb{U} : Ax \dot{+} Bu \in \mathcal{P}\}, \quad \forall \mathcal{P} \subseteq \mathbb{R}^n. \quad (7)$$

If $\mathcal{P}$ and $\mathbb{U}$ are polytopes, then $Pre(\mathcal{P})$ 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 $\mathcal{P}_q$ reaches $\mathcal{P}_{q'}$. The computational aspects of this cost are presented in Section V. 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 $\mathcal{A}^D$ using Dijkstra's algorithm [26]. The $Partitioning$ procedure, which will be presented in detail in the next subsection, partitions a state region and modifies $\mathcal{A}^D$ accordingly.

The $FeasibilityTest(q,q')$ procedure checks if there exists a controller that enables (q,q') and returns the cost $J((q,q'))$. 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 $\mathcal{P}_q$ to the beacon of (q,q'), i.e., $\mathcal{B}_{qq'}$, in finite time without leaving the set $\mathcal{P}_q$. Notice that to solve the $\mathcal{P}_q$-to-$\mathcal{P}_{q'}$ problem it suffices to solve the $\mathcal{P}_q - to - \mathcal{B}_{qq'}$ problem, since a trajectory originating in $\mathcal{P}_q$ will reach $\mathcal{P}_{q'}$ without leaving $\mathcal{P}_q$ only through the beacon $\mathcal{B}_{qq'}$. By definition, there exists an admissible control action for all $x \in \mathcal{B}_{qq'}$ such that $\mathcal{P}_{q'}$ is reached in one step. If q does not have a self transition, the transition (q,q') is only enabled when $\mathcal{P}_q = \mathcal{B}_{qq'}$, since $\mathcal{B}_{qq'}$ is the largest set of states in $\mathcal{P}_q$ that can reach $\mathcal{P}_{q'}$ in one step.

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 3 and 10). Then, a state q_s is selected from the set of candidate states for partitioning by considering the path costs in line 6.

Partitioning: A state q is partitioned into a set of states $\{q_1, \ldots, q_d\}$ via a polytopic partition of $\mathcal{P}_q$. 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

Input: Dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, 2^P, \tau^D, Q_0^D, F^D)$, a state $q_P \in Q^D$

Output: Partitioned automaton $\mathcal{A}^{D'} = (Q^{D'}, \rightarrow^{D'}, 2^P, \tau^{D'}, Q_0^{D'}, F^D)Q^P = \{q_1, \ldots, q_d \mid \mathcal{P}_{q_P} = \cup_{i=1,\ldots,d}\mathcal{P}_{q_i}, and\ \mathcal{P}_{q_i} \cap \mathcal{P}_{q_j} = \emptyset\ if\ i \neq j\}$

1:$Q^P = \{q_1, \ldots, q_d \mid \mathcal{P}_{q_P} = \cup_{i=1,\ldots,d}\mathcal{P}_{q_i}$, and $\mathcal{P}_{q_i} \cap \mathcal{P}_{q_j} = \emptyset\ if\ i \neq j\}$

2:$Q^{D'} := (Q^D \setminus \{q_P\}) \cup Q^P$

3:$\rightarrow^{D'} = \rightarrow^D \setminus \{\{(q, q_P) \mid q \in Q^D\} \cup \{(q_P, q) \mid q \in Q^D\}\}$

4:$\tau^{D'}(q) = \tau^D(q)$ for each $q \in Q^D \setminus \{q_P\}$, and $\tau^{D'}(q_i) := \tau^D(q_P)$ for each $q_i \in Q^P$

5:$Q_0^{D'} = Q_0^D \setminus \{q_P\}$

6:$Q_0^{D'} = Q_0^{D'} \cup Q^P$ if $q_P \in Q_0^D$

7:$\rightarrow^{D'} := \rightarrow^{D'} \cup \{(q, q_i) \mid (q, q_P) \in \rightarrow^D, q_i \in Q^P, Post(q) \cap \mathcal{P}_{q_i} \neq \emptyset\}$

8:$\rightarrow^{D'} := \rightarrow^{D'} \cup \{(q_i, q) \mid (q_P, q) \in \rightarrow^D, q_i \in Q^P, Post(q_i) \cap \mathcal{P}_q \neq \emptyset\}$

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 $\mathcal{B}_{qq'}$. Notice that beacons will always be polytopes, as $Pre(\mathcal{P}_{q'})$ 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 $\mathcal{B}_{qq'}$.

Let $\mathcal{A}^{D_i} = (Q^{D_i}, \rightarrow^{D_i}, 2^P, \tau^{D_i}, Q_0^{D_i}, F^{D_i})$ denote the dual automaton after refinement iteration i, and let $\mathcal{A}^{D_0}$ denote the initial dual automaton. For a dual automaton $\mathcal{A}^{D_i}$, the set $\mathbb{X}_{0,i}^\Phi \subseteq \mathbb{X}$ denotes the union of the regions corresponding to start states of automaton $\mathcal{A}^{D_i}$ with finite path costs, i.e.

$$\mathbb{X}_{0,i}^\Phi := \bigcup_{q \in \{q' \in Q_0^{D_i} \mid J^P(q') < \infty\}} \mathcal{P}_q \subseteq \mathbb{X}. \tag{8}$$

The refinement algorithm (Alg. 2) 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. Note that when a state is partitioned, the new states inherit only the transitions that satisfy a reachability condition (see lines 7 and 8 of Alg. 3). Therefore, when the algorithm stops either all the states have finite costs or the regions associated with states that have finite costs are not reachable from the regions associated with states that have infinite costs through automaton transitions.

Example IV.2: 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 Ex. IV.1 [Note that the automata in Fig. 1(b) and in Fig. 3(a) represent $\mathcal{A}^{D_0}$. For simplicity the final state labeled by T is not shown in Fig. 3(a)]. $\mathcal{A}^{D_0}$ 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 $\mathcal{B}_{q_1 q_2}$ generates the state regions and the automaton shown in Fig. 3(c) and (d). As $Post(\mathcal{P}_{q_3}) \cap \mathcal{P}_{q_2} = \emptyset$, the transition (q_3, q_2) is not added. In the next iteration q_1 is partitioned using $\mathcal{B}_{q_1 q_2}$ 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 Section V were used to check the costs of the transitions.

Proposition IV.3: Assume $\mathbb{X}_{0,0}^\Phi$ is non-empty. Given an arbitrary iteration $i \geq 1$ of Alg. 2, the set $\mathbb{X}_{0,i}^\Phi$ as defined in Equation (8) has the following properties:

(i) There exists a sequence of admissible control actions such that every closed-loop trajectory of system (2) originating in $\mathbb{X}_{0,i}^\Phi$ satisfies the formula Φ, and

(ii) $\mathbb{X}_{0,i-1}^\Phi \subseteq \mathbb{X}_{0,i}^\Phi$.

Proof: (1) A finite path cost for a state $q_0 \in Q_0^{D_i}$ implies that there exists an automaton run $q_0 \ldots q_d$ with $J((q_j, q_{j+1})) < \infty$ for all $j = 0, \ldots, d-1$ 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 $\mathcal{P}_{q_0}$ reaches $\mathcal{P}_{q_d}$ by following the automaton path. Considering that removing states and transitions only reduces the language of the automaton, by Prop. II.5 it follows that $\mathcal{L}_{\mathcal{A}^{D_0}} \subseteq \mathcal{L}_\Phi$. Since the proposed partitioning procedure preserves the language, we have $\mathcal{L}_{\mathcal{A}_i^D} \subseteq \mathcal{L}_{\mathcal{A}^{D_{i-1}}}$. Consequently, $\mathcal{L}_{\mathcal{A}^{D_i}} \subseteq \mathcal{L}_\Phi$ and the resulting trajectories satisfy the formula.

(ii) For any $x \in \mathbb{X}_{0,i-1}^\Phi$, there exists an accepting automaton run $r_D = q_0 \ldots q_d$ with $x \in \mathcal{P}_{q_0}$ 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 $j = 0, \ldots, d$ as $J^P(q_j) < \infty$ for all $j = 0, \ldots, d$. As only the transitions adjacent to q_s are affected by partitioning, $q_j \in Q_i^D$ for all $j = 0, \ldots, d$ and $(q_j, q_{j+1}) \in \rightarrow^{D_i}$ for all $j = 0, \ldots, d-1$. Therefore, $x \in \mathcal{P}_{q_0}$, $r_D = q_0 \ldots q_d$ is an accepting run of $\mathcal{A}^{D_i}$ with finite cost

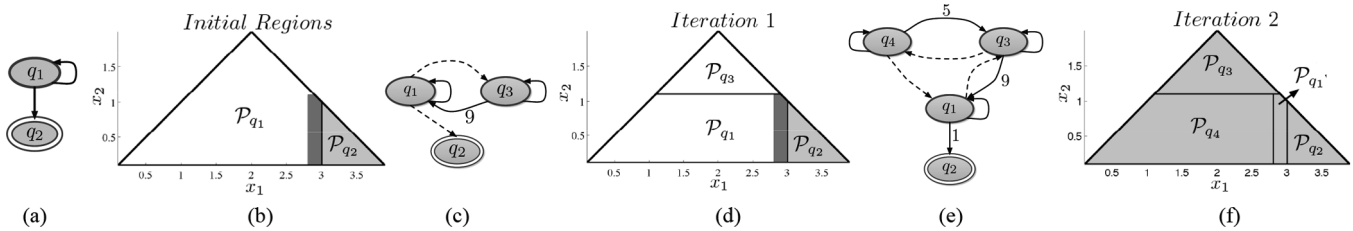


Fig. 3. Automata and their corresponding polytopic state-space partitions for the iterations of Ex. IV.2. The polytopes are shown with black borders and $\mathbb{X}_{0,i}^{\Phi_1}$ is shown in light gray. The beacon of transition (q_1, q_2) is shown in dark gray in Figs. (b) and (d). A transition is shown with dashed line if a controller that enables the transition was not found, otherwise the transition is marked with a time bound.

and thus, $x \in \mathbb{X}_{0,i}^{\Phi}$. Observing that $x \in \mathbb{X}_{0,i-1}^{\Phi}$ was chosen arbitrarily completes the proof. ∎

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.

V. Polytope-to-Polytope Control

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

Problem V.1: Let $\mathcal{B}$ and $\mathcal{P}$ be polytopes in $\mathbb{R}^n$ with $\mathcal{B} \subseteq \mathrm{Pre}(\mathcal{P})$ and consider system (2). Construct a state-feedback control law $g : \mathbb{R}^n \to \mathbb{R}^m$ such that

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

1) Problem V.2: Let $N \in \mathbb{Z}_+$, and $N \geq 1$, $\mathcal{B}$ and $\mathcal{P}$ be polytopes in $\mathbb{R}^n$ with $\mathcal{B} \subseteq \mathcal{P}$ and consider system (2). Construct a state-feedback control law $g : \mathbb{R}^n \to \mathbb{R}^m$ such that for all $x_0 \in \mathcal{P}$

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

Notice that while Prob. V.1 is always feasible since $\mathcal{B} \subseteq \mathrm{Pre}(\mathcal{P})$, Prob. V.2 needs not be feasible for any set $\mathcal{P}$ and corresponding beacon $\mathcal{B}$. In what follows, sufficient conditions for feasibility of Prob. V.2 will be given.

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

$$\begin{aligned} H_{\mathcal{P}}(Av^i + Bu^i) &\leq h_{\mathcal{P}}, \\ H_{\mathbb{U}}u^i &\leq h_{\mathbb{U}}. \end{aligned} \tag{9}$$

A solution to (9) can be obtained *off-line* by solving a feasibility LP. It is trivial to deduce that the control law

$$g(x) := \sum_{i=1}^{\#\mathcal{V}(\mathcal{B})} \lambda_i u^i \tag{10}$$

where $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, are such that $x = \sum_{i=1}^{\#\mathcal{V}(\mathcal{B})} \lambda_i v^i$, solves Prob. V.1. The evaluation of the control law (10) requires *on-line* calculation of the coefficients $\{\lambda_i\}_{i=1,\dots,\#\mathcal{V}(\mathcal{B})}$, which amounts to solving a system of linear equations and can also be formulated as a feasibility LP.

Alternatively, an explicit PWA form of $g(\cdot)$ can be obtained by a simplicial partition of $\mathcal{B}$. Then, the evaluation of g requires solving *on-line* a point location problem [27], 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 $\{\lambda_i\}_{i=1,\dots,\#\mathcal{V}(\mathcal{B})}$ *on-line* . Yet another explicit PWA solution to Prob. V.1 can be obtained via direct synthesis of a PWA control law defined over an arbitrary polytopic partition of $\mathcal{B}$, 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. V.2, i.e., vertex interpolation and contractive sets.

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

$$\begin{aligned} x_0^i &:= v^i, \\ x_{k+1}^i &= Ax_k^i + Bu_k^i, \quad \forall k = 0, \dots, N-1, \\ H_{\mathcal{P}}x_k^i &\leq h_{\mathcal{P}},\ H_{\mathbb{U}}u_k^i \leq h_{\mathbb{U}}, \quad \forall k = 0, \dots, N-1, \\ H_{\mathcal{B}}x_N^i &\leq h_{\mathcal{B}}. \end{aligned} \tag{11}$$

A solution to the set of problems (11) can be searched for off-line by solving repeatedly a corresponding set of feasibility LPs starting with $N = 1$, for all $i = 1, \dots, \#\mathcal{V}(\mathcal{P})$, 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 $x \in \mathcal{P}$, the control law

$$g(x_k) := \sum_{i=1}^{\#\mathcal{V}(\mathcal{P})} \lambda_i u_k^i, \quad k = 0, \ldots, N^* - 1 \tag{12}$$

where $x_0 = x$ and $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, are such that $x = \sum_{i=1}^{\#\mathcal{V}(\mathcal{P})} \lambda_i v^i$, solves Prob. V.2 and yields that closed-loop trajectories reach $\mathcal{B}$ in at most N^* discrete-time instants.

Evaluation of the control law g of (12) at time $k = 0$ requires *on-line* calculation of the coefficients $\{\lambda_i\}_{i=1,\ldots,\#\mathcal{V}(\mathcal{P})}$, which is a LP, while at every $k = 1, \ldots, N^*-1$ the analytic expression of g is implemented. However, a faster convergence to $\mathcal{B}$ can be obtained by taking $\lambda_i \in \mathbb{R}$, $0 \leq \lambda_i \leq 1$, such that $x = \sum_{i=1}^{\#\mathcal{V}(\mathcal{P})} \lambda_i x_{j^*}^i$, where

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

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

Similarly as in the case of the control law (10), simplicial decompositions of $\mathcal{P}$ can be employed to obtain an explicit PWA form of the control law $g(x_k)$, $k = 0, \ldots, N^* - 1$, both for its standard and faster variants presented above.

Remark V.3: In general, existence of a finite, common N such that all LPs (11) 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 $\mathcal{P}$. In the "worst" case, the partitioning converges to the maximal controllable subset of $\mathcal{P}$ with respect to $\mathcal{B}$, 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 $\mathcal{P}$, which can result in a significant complexity reduction, as it is illustrated for the case studies presented in Section VIII.

Sufficient conditions for feasibility of the LPs (11) can be obtained as follows. Let

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

If $\mathcal{E}_{\mathcal{B}} \neq \emptyset$ and the polytope $\mathcal{P}$ induces [28] a local control Lyapunov function for system(2) with respect to an equilibrium point[3] $x^s \in \mathcal{E}_{\mathcal{B}}$, there always exists a $N_i \geq 1$ such that the LPs (11) are feasible for all $i = 1, \ldots, \#\mathcal{V}(\mathcal{P})$. 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 $x^{s,i} \in \mathcal{E}_{\mathcal{B}}$. Notice that the same assumptions were employed in [21], where only obstacle avoidance specifications were considered and polyhedral Lyapunov functions were used. In this respect, the proposed vertex interpolation solution for solving Prob. V.2 can be regarded as a relaxation of standard interpolation synthesis methods, where existence of a closed-loop equilibrium is assumed.

Contractive Sets Pick any $\bar{x} \in \mathrm{int}(\mathcal{B})$ (not necessarily an equilibrium point) and let $\mathcal{M}(x) := \max_{i=1,\ldots,w} W_{i\bullet}(x - \bar{x})$ denote a function induced by the polytope $\mathcal{P}$ and the point $\bar{x}$, where $w \geq n + 1$ is the number of lines of the matrix W, which is such that $\mathcal{P} = \{x \in \mathbb{R}^n \mid \mathcal{M}(x) \leq 1\}$. Moreover, let $\alpha_{\bar{x}}^\star = \max_{\alpha>0}\{\alpha(\mathcal{P} \oplus \{-\bar{x}\}) \subseteq (\mathcal{B} \oplus \{-\bar{x}\})\}$. Note that $\alpha_{\bar{x}}^\star < 1$ whenever $\mathcal{B} \subseteq \mathcal{P}$ and $\mathcal{B} \neq \mathcal{P}$; pick any $\alpha_{\bar{x}}$ such that $0 < \alpha_{\bar{x}} < \alpha_{\bar{x}}^\star$. Next, consider the conic polytopic partition $\{\mathcal{C}_i\}_{i=1,\ldots,w}$ of $\mathcal{P}$ induced by $\bar{x}$, which is constructed as follows:

$$\mathcal{C}_i := \{x \in \mathcal{P} | (W_{i\bullet} - W_{j\bullet})(x - \bar{x}) \geq 0,\ j = 1, \ldots, w,\ j \neq i\}.$$

Notice that $\cup_{i=0,\ldots,w} \mathcal{C}_i = \mathcal{P}$ and $\mathrm{int}(\mathcal{C}_i) \cap \mathrm{int}(\mathcal{C}_j) = \emptyset$ 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

$$g(x) := K_i x + a_i \quad \text{if} \quad x \in \mathcal{C}_i \tag{13}$$

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

$$\begin{aligned} \rho(W_{i\bullet}(x - \bar{x}) - \alpha_{\bar{x}}) - (W_{j\bullet}(Ax + Bg(x) - \bar{x}) - \alpha_{\bar{x}}) \geq 0, \\ \forall x \in \mathcal{V}(\mathcal{C}_i), \forall j = 1, \ldots, w, \forall i = 1, \ldots, w, \\ K_i x + a_i \in \mathbb{U}, \quad \forall x \in \mathcal{V}(\mathcal{C}_i), \forall i = 1, \ldots, w. \end{aligned} \tag{14}$$

Notice that ρ can be minimized to obtain an optimal convergence rate and a different ρ_i can be assigned to each cone $\mathcal{C}_i$, while (14) remains a LP. If the LP (14) is feasible, then by construction and the definition of $\mathcal{M}(\cdot)$, for all $x \in \mathcal{P}$ it holds that

$$\rho\mathcal{M}(x) - \mathcal{M}(Ax + Bg(x)) \geq \alpha_{\bar{x}}(\rho - 1). \tag{15}$$

The recursive application of (15) implies that for a closed-loop trajectory $\{x_k\}_{k\geq 0}$, the following inequality holds for all $k \geq 0$:

$$\mathcal{M}(x_k) \leq \rho^k \mathcal{M}(x_0) - \alpha_{\bar{x}}\rho^k + \alpha_{\bar{x}}. \tag{16}$$

The above property can be exploited to establish the following result.

Lemma V.4: Suppose that the LP (14) is feasible and let

$$k^\star := \left\lceil \frac{\ln\left(\frac{\alpha_{\bar{x}}^\star - \alpha_{\bar{x}}}{1 - \alpha_{\bar{x}}}\right)}{\ln(\rho)} \right\rceil. \tag{17}$$

Then for all trajectories $\{x_i\}_{i\geq 0}$ with $x_0 \in \mathcal{P}$ of system (2) in closed-loop with(13) there exists a $k \geq 0$ such that $x_k \in \mathcal{B}$ and, moreover, $k \leq k^\star$.

Proof: Let $x_0 \in \mathcal{P}$. To prove the claim consider two cases, i.e., $\mathcal{M}(x_0) \leq \alpha_{\bar{x}}^\star$ and $\mathcal{M}(x_0) > \alpha_{\bar{x}}^\star$. Notice that for any $x \in \mathcal{P}$, $\mathcal{M}(x) \leq \alpha_{\bar{x}}^\star$ implies that $x \in (\alpha_{\bar{x}}^\star(\mathcal{P} \oplus \{-\bar{x}\}) \oplus \{\bar{x}\})$ and thus, it implies that $x \in \mathcal{B}$. As such, in the case when $\mathcal{M}(x_0) \leq \alpha_{\bar{x}}^\star$, the claim holds by the definition of $k^\star$ (Equation (17)).

Next, consider the case when $\mathcal{M}(x_0) > \alpha_{\bar{x}}^\star$ and suppose that the inequality

$$\alpha_{\bar{x}}^\star < \rho^k \mathcal{M}(x_0) - \alpha_{\bar{x}}\rho^k + \alpha_{\bar{x}} \tag{18}$$

holds for all $k \geq 0$. By taking the limit when k tends to infinity in the above inequality yields that $\alpha_{\bar{x}}^\star < \alpha_{\bar{x}}$ and thus, a contradiction was reached. As such, there must exist a $k \geq 0$ such that

$$\rho^k \mathcal{M}(x_0) - \alpha_{\bar{x}}\rho^k + \alpha_{\bar{x}} \leq \alpha_{\bar{x}}^\star \tag{19}$$

[3]When $x^s = 0$, the function induced by $\mathcal{P}$ is called the Minkowski or gauge function of $\mathcal{P}$[28]. For Lyapunov functions induced by sets that do not contain the origin we refer the interested reader to [29].

holds. Equation (16) and(19) imply that $\mathcal{M}(x_k) \leq \alpha^\star_{\bar{x}}$ and hence, $x_k \in \mathcal{B}$. Furthermore, reordering the terms and taking the logarithm of both sides in Equation(19) yields

$$k \leq \frac{\ln\left(\frac{\alpha^\star_{\bar{x}} - \alpha_{\bar{x}}}{\mathcal{M}(x_0) - \alpha_{\bar{x}}}\right)}{\ln(\rho)}. \tag{20}$$

Noticing that $\mathcal{M}(x) \leq 1$ for all $x \in \mathcal{P}$, the fact that $k \leq k^\star$ follows directly from the definition of $k^\star$ (Equation (17)). As the point $x_0 \in \mathcal{P}$ was chosen arbitrarily, the claim was established. ∎

The above result establishes that if the LP (14) is feasible, the PWA control law(13) is an admissible solution to Prob. V.2. The *on-line* evaluation of (13) reduces to a point location problem that can be solved in logarithmic time due to the specific conic partition.

Remark V.5: Notice that if $\bar{x}$ is an equilibrium point, then by augmenting the constraints of the LP (14) with the equilibrium point constraints, i.e., $(A + BK_i)\bar{x} + Ba_i = \bar{x}$, for all $i = 1, .., w$, and setting $\alpha_{\bar{x}}$ to 0 we recover the solution proposed in [22] and the function $\mathcal{M}(\cdot)$ becomes a standard Lyapunov function with respect to the equilibrium $\bar{x}$.

VI. Complete Control Strategy

In this section, we provide a solution to Prob. III.1 by using the results from Sections IV and V. The proposed control strategy that solves Prob. III.1 is composed of a finite state dual automaton $\mathcal{A}^C$ (see Def. II.4) and a map M from transitions of $\mathcal{A}^C$ to state feedback controllers. The control strategy $(\mathcal{A}^C, M)$ is constructed as defined in Def. VI.1 from the dual automaton $\mathcal{A}^{D_R}$ and the cost functions J and J^P obtained from Alg. 2. The state feedback controllers assigned by M are constructed as described in Section V. Existence of these controllers is guaranteed, since $\mathcal{A}^C$ has only finite cost transitions.

Definition VI.1 (Control Strategy): Given a dual automaton $\mathcal{A}^{D_R} = (Q^{D_R}, \rightarrow^{D_R}, 2^P, \tau^{D_R}, Q_0^{D_R}, F^{D_R})$, and cost functions $J : \rightarrow^{D_R} \longrightarrow \mathbb{Z}_+$ and $J^P : Q^{D_R} \longrightarrow \mathbb{Z}_+$, the control strategy is a pair $(\mathcal{A}^C, M)$, where $\mathcal{A}^C = (Q^C, \rightarrow^C, 2^P, \tau^C, Q_0^C, F^C)$ is a dual automaton defined as

$$\begin{aligned}
&Q^C = \{q \in Q^{D_R} \mid J^P(q) < \infty\},\\
&\rightarrow^C = \{(q, q') \mid J((q, q')) < \infty, (q, q') \in \rightarrow^{D_R}, q' \in Q^C\},\\
&\tau^C : Q^C \rightarrow 2^P, \quad \tau^C(q) = \{p \mid [p] \cap \mathcal{P}_q \neq \emptyset\},\\
&Q_0^C = Q_0^D \cap Q^C,\\
&F^C = F^{D_R}
\end{aligned}$$

and M is a map from transitions of $\mathcal{A}^C$ to corresponding state feedback controllers.

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

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

$$\mathbb{X}_0^\Phi = \bigcup_{q \in Q_0^C} \mathcal{P}_q. \tag{21}$$

For a given accepting run $r_C = q_0 \dots q_d$ of $\mathcal{A}^C$, the time required to satisfy the specification for trajectories originating in $\mathcal{P}_{q_0}$ 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 $x_0 \in \mathbb{X}_0^\Phi$ of system (2) is upper bounded by $\max_{q_0 \in Q_0^C} J^P(q_0)$.

Proposition VI.2: Every closed-loop trajectory produced by the control strategy $(\mathcal{A}^C, M)$ satisfies the formula Φ.

Proof: The proof that all the trajectories of the closed loop system satisfy the formula follows immediately from Prop. IV.3 since $\mathbb{X}_0^\Phi = \mathbb{X}_{0,D_R}^\Phi$. ∎

The following theorem states that when the refinement algorithm terminates, the proposed solution to Prob. III.1 is complete.

Theorem VI.3: Suppose Alg. 2 terminates. Then any trajectory of system(2) that produces a word $w \in \mathcal{L}_\Phi$ originates in $\mathbb{X}_0^\Phi$, i.e., $\mathbb{X}_0^\Phi = \overline{\mathbb{X}_0^\Phi}$.

Proof: To show that any satisfying trajectory originates in $\mathbb{X}_0^\Phi$, assume by contradiction that there exists $x_0 \notin \mathbb{X}_0^\Phi$ such that $x_0 \dots x_d$ is a satisfying trajectory of system (2), i.e., $P_0 \dots P_d \in \mathcal{L}_\Phi$. Then by Prop. II.5, there exists an accepting run $r_D = q_0 \dots q_d$ of the initial dual automaton $\mathcal{A}^{D_0}$ such that $x_k \in \mathcal{P}_{q_k}, \forall k = 0, \dots, d$. The run r_D induces a unique refined dual automaton run $r_{D^R} = q'_0 \dots q'_d$ where q'_k and q_k coincide or q'_k is obtained from q_k through partitioning and $x_k \in \mathcal{P}_{q'_k} \subseteq \mathcal{P}_{q_k}$ for all $k = 0, \dots, d$.

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

$$Post(\mathcal{P}_q) \bigcap \left\{ \bigcup_{J^P(q') < \infty, (q,q') \in \rightarrow^{D^R}} \mathcal{P}_{q'} \right\} = \emptyset \tag{22}$$

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

As shown in Prop. IV.3, 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. VI.3 states that when this iteration converges in finite time then the maximal set of satisfying states has been obtained, i.e.,

$\mathbb{X}_0^\Phi = \overline{\mathbb{X}_0^\Phi}$. This is possible whenever the maximal set is a polytope or a union of polytopes, since $\mathbb{X}_0^\Phi$ is defined as union of polytopes and we obtain one-step controllable sets partition of each state region in the worst case. As this is not necessarily the case for an arbitrary 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). Notice that stopping Alg. 2 at any iteration $i \geq 1$ will still result in a set of initial *satisfying* states by Prop. IV.3. As such, indeed, Alg. 2 can be stopped whenever a desired subset of the state-space has been covered.

A. Complexity

The complexity of the proposed solution can be analyzed in two aspects: off-line and on-line parts. The off-line part involves the construction of the dual automaton, the simplification routines, and the refinement algorithm. The construction of an FSA from a co-safe LTL formula Φ leads to a double exponential blowup [24], therefore the dual automaton $\mathcal{A}^D$ has $2^{2^{O(|\Phi|)}}$ transitions. However, this theoretical bound is not usually achieved in practice [25]. Moreover, the compact representation, the DNF simplification and the pruning algorithm significantly reduce the automaton size (see Table I for experimental results). In Alg. 1, one step reachable sets are computed for each state to check the feasibility of transitions, which requires $|Q^D| + |\rightarrow^D|$ basic polyhedral operations [see (6)]. Then, the feasibility of the states are checked by traversing the underlying graph, which is polynomial in $|Q^D|$.

The complexity of Alg. 2 is dictated by the number of iterations $R \in \mathbb{Z}_+$, which depends on the control system (2), the specification, and the employed polytope-to-polytope controller synthesis method. Initially, transition (J) and state (J^P) costs are computed via $FeasibilityCheck$ and Dijkstra's algorithm, respectively. The run time of Dijkstra's algorithm is $O(|Q^{D_0}|^2)$. $FeasibilityCheck$ procedure involves basic polyhedral operations (see (7)) and controller synthesis, i.e., solving Prob. V.1 and Prob. V.2. The proposed controller synthesis methods are based on linear programming. The number of constraints of the vertex interpolation-based solutions depend exponentially on the size of the state space, whereas, the number of constraints of the contractive sets method depends polynomially on the size of the state space. At each iteration of Alg. 2, a candidate state is partitioned into two states, and $FeasibilityCheck$ procedure is used to compute the costs of the new transitions. Let d be the maximum number of transitions incident to a state $q \in Q^{D_i}$ for all iterations $i = 0, \ldots, R$. Then, the total number of $FeasibilityCheck$ procedures run is upper bounded by $|\rightarrow^{D_0}| + 2dR$. Note that, at each iteration it is sufficient to compute costs (J^P) for the new states and the states that can reach these states.

The on-line part deals with the generation of the control input for system (2) and involves linear programming.

VII. Extensions

In this section, we discuss some extensions of the approach presented above to more general system dynamics and specifications. A quick examination of the algorithms shows that the method works if the regions $\mathcal{P}$ in the partitions are polytopes, their pre-images $Pre(\mathcal{P})$ through the system dynamics are computable, the beacons $\mathcal{B}$ of transitions are polytopes, and the polytope-to-polytope controllers can be computed. Discrete-time PWA systems with polytopic partitions satisfy these conditions. In Section VII-A, we discuss the details of this extension. To accommodate richer temporal logic specifications, the refinement algorithm (Alg. 2) needs to be adapted to different types of automaton acceptance conditions. In Section VII-B, the extension to full LTL is explained briefly.

A. Extension to PWA Systems

A discrete-time PWA control system is described by

$$x_{k+1} = A_i x_k + B_i u_k + c_i, x_k \in \mathbb{X}_i,\ u_k \in \mathbb{U}_i,\ i = 1, \ldots, l \quad (23)$$

where l is the number of modes (different dynamics), $\mathbb{X}_i$, $i = 1, \ldots, l$ provide a polytopic partition of $\mathbb{X} = \bigcup_{i=1,\ldots,l} \mathbb{X}_i \subset \mathbb{R}^n$, $\mathbb{U}_i$ are arbitrary polytopes in $\mathbb{R}^m$, and A_i, B_i, and c_i are matrices of appropriate sizes.

The extension of the method from the linear dynamics (2) to the PWA dynamics (23) requires an additional refinement procedure that takes into account the particular structure of the state space $\mathbb{X} = \bigcup_{i=1,\ldots,l} \mathbb{X}_i$. This procedure is summarized in Alg. 4.

Alg. 4 preserves the dual automaton language while ensuring that for each dual automaton state q there exists $i \leq l$ such that $\mathcal{P}_q \subseteq \mathbb{X}_i$. Consequently, for each dual automaton state q there exist an associated system dynamics (A_i, B_i, c_i) and an admissible set of controls $\mathbb{U}_i$ that can be used to compute the beacons $\mathcal{B}_{qq'}$ of transitions leaving the state q and $Post(\mathcal{P}_q)$, and to solve controller synthesis problems of type Prob. V.1 and Prob. V.2 Therefore, the algorithms proposed in Section IV with control synthesis tools described in Section V can directly be used for PWA systems after the refinement step. Also, it is easy to see that the completeness of the solution for PWA systems is guaranteed under the same finite time termination assumption of Alg. 2 as in Thm VI.3. The extension to PWA systems is illustrated for an example inspired by systems biology in Section VIII.

Algorithm 4 PWA Refinement

Input: Dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, 2^P, \tau^D, Q_0^D, F^D)$, and regions $\mathbb{X} = \bigcup_{i=1,\ldots,l} \mathbb{X}_i$

Output: Refined automaton $\mathcal{A}^{D'} = (Q^{D'}, \rightarrow^{D'}, 2^P, \tau^{D'}, Q_0^{D'}, F^{D'})$

1:**for all** $q \in Q_D$ **do**

2:$q_Q := \{q_j \mid \mathcal{P}_{q_j} = \mathcal{P}_q \cap \mathbb{X}_j, \mathcal{P}_{q_j} \neq \emptyset\}$

3:$Q_{D'} = Q_{D'} \cup q_Q$

4:$\rightarrow^{D'} := \rightarrow^{D'} \cup \left\{\bigcup_{(q,q') \in \rightarrow^D} \{(q_j, q') \mid q_j \in q_Q\}\right\}$
$\cup \left\{\bigcup_{(q',q) \in \rightarrow^D} \{(q', q_j) \mid q_j \in q_Q\}\right\}$

5:$Q_0^{D'} := Q_0^{D'} \cup q_Q$ if $q \in Q_0^D$

6:$F^{D'} := F^{D'} \cup q_Q$ if $q \in F^D$

7:$\tau^{D'}(q_j) = \tau^D(q)$ for all $q_j \in q_Q$

8:**end for**

TABLE I
CASE STUDY SUMMARY

$Case\#$	$\mathcal{A}, \lvert Q\rvert$	$\mathcal{A}, \lvert \rightarrow \rvert$	$\lvert C\rvert$	$\mathcal{A}^D, \lvert Q^D\rvert$	$\mathcal{A}^D, \lvert \rightarrow^D \rvert$	$\mathcal{A}^{D_0}, \lvert Q^{D_0}\rvert$	$\mathcal{A}^{D_0}, \lvert \rightarrow^{D_0} \rvert$	$\mathcal{A}^C, \lvert Q^C\rvert$	$\mathcal{A}^C, \lvert \rightarrow^C \rvert$	$\lvert Ic\rvert$	t
1	3	6	2425	72	2452	69	531	228	879	183	$89sec.$
2	2	3	21	16	225	16	124	623	840	902	$48min.$
3	2	4	43	12(32)	121(961)	32	165	107(32)	418(165)	75(0)	$30sec.(13sec.)$

The number of states and transitions of the FSA $\mathcal{A}$ that accepts the specification formula, the initial dual automaton $\mathcal{A}^D$, the pruned dual automaton $\mathcal{A}^{D_0}$, and the automaton $\mathcal{A}^C$ for each case study. The number of conjunctive clauses deleted by the DNF simplification, the number of iterations of Alg. 2, and the computation times are denoted by $|C|$, $|Ic|$, and t, respectively. The size of the dual automaton obtained from Alg. 4 and the results, when the contractive sets method is employed, are given in parenthesis for Case Study 3.

B. Extension to Full LTL

To extend the set of allowed specifications from scLTL to full LTL, the main challenge is to extend our approach from the acceptance condition of an FSA to the more complicated acceptance condition of a Büchi automaton. There exist algorithms for the construction of such an automaton that accepts the language satisfying an arbitrary LTL formula [1]. The extension involves the solution of a backward reachability problem, which enforces the Büchi acceptance condition. Briefly, any run accepted by a Büchi automaton has a prefix-suffix structure [30]. The suffix is an infinite Büchi automaton run that periodically visits some accepting state. Starting from this observation, the scLTL approach can be extended to full LTL in two steps. First, the refinement algorithm (Alg. 2) can be used to find a cycle originating from an accepting state of the Büchi automaton. When a cycle is found, the algorithm can be used to find the set of states that can reach the corresponding accepting state. The full LTL extension is not implemented in the software package.

A limitation of this extension is that completeness cannot be guaranteed. Since only a single cycle is used, the initial states that can reach another cycle are not taken into consideration. Our proposed solution provides the maximum time to reach the accepting state from the cycle (for prefix), and the maximum time between consecutive visits of an accepting state (for suffix).

VIII. IMPLEMENTATION AND CASE STUDIES

The computational framework developed in this paper was implemented as a Matlab software package LanGuiCS (Language-Guided Control Synthesis), which is freely downloadable from hyness.bu.edu/software. The toolbox takes as input an scLTL formula over a set of linear predicates, the matrices of a PWA system, control constraints sets and operating regions, and produces a solution to Prob. III.1 in the form of a set of initial states and a state-feedback control strategy. The tool, which uses *scheck2* [25] for the construction of the FSA, MPT [31] for polyhedral operations, and Gurobi [32] for solving LPs, also allows for displaying the set of initial states and simulating the trajectories of the closed-loop system for 2-D or 3-D state-spaces.

The three case studies described below were generated using LanGuiCS running on an iMac with a Intel Core i5 processor at 2.8 GHz with 8 GB of memory. Some illustrative numbers including the computation times are summarized in Table I. Note that in all case studies the refinement algorithm (Alg. 2) terminated in a finite number of steps, which implies that the set of satisfying initial states was maximal.

A. Case Study 1 : Double Integrator

Obstacle avoidance for double integrators is a particularly challenging problem [20]. The double integrator dynamics with sampling time of 1 sec. are of the form in Equation (2), where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}. \tag{24}$$

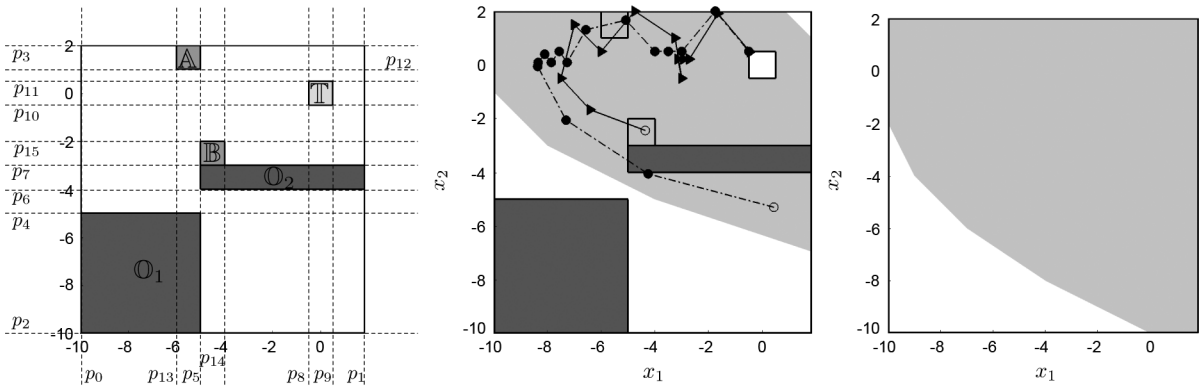
We assume that the control constraint set is $\mathbb{U} = \{u \mid -2 \leq u \leq 2\}$. The specification is to visit region $\mathbb{A}$ or region $\mathbb{B}$, and then the target region $\mathbb{T}$, while always avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, and staying inside a safe region given by $\mathbb{X} = \{x \mid -10 \leq x_1 \leq 1.85, -10 \leq x_2 \leq 2\}$. The sets $\mathbb{X}, \mathbb{U}$ and the obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$ are the same as the ones used in [20], where the goal was to synthesize a control strategy such that a trajectory of a double integrator with disturbance originating in a given state $x_0 \in \mathbb{X}$ reaches the target while avoiding obstacles. All these polytopic regions, together with the linear predicates used in their definitions, are shown in Fig. 4(a). Using these predicates, the specification can be written as the following scLTL formula:

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

The computation summary for this case study is given in the first row of Table I. The set $\mathbb{X}_0^{\Phi_2}$ and a sample of satisfying trajectories of the closed loop system are shown in Fig. 4(b). Every trajectory originating in $\mathbb{X}_0^{\Phi_2}$ satisfies the specification within 20 discrete-time instants. The polytope-to-polytope controllers are synthesized using vertex interpolation.

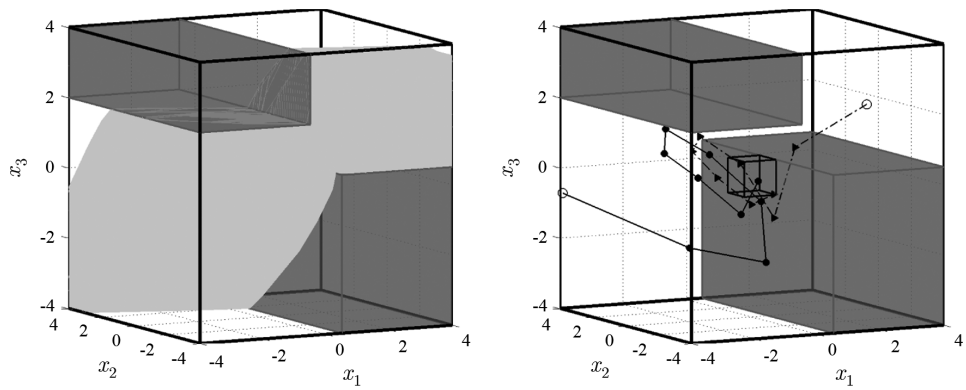
As discussed in the paper, the upper time bound is affected by the choice of candidate polytopes for partitioning. In this example, a transition is selected from the candidate set according to the cost of the target state as described in Alg. 2. Our experiments showed that choosing the transition corresponding to the beacon with the highest Chebychev ball radius resulted in a faster coverage (94 iterations). However, it also produced a higher time bound of 43 steps.

For the double integrator dynamics (24), 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 shown in Fig. 4(c) for the dynamics(24) 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. V.2. The computation of the actual maximal set can easily be verified by iteratively computing one-step controllable sets within $\mathbb{X}$ starting from an invariant polytopic set $\mathcal{P}_0$ which contains the origin, i.e, $\mathcal{P}_n = Pre(\mathcal{P}_{n-1}) \cap \mathbb{X}, 0 \in \mathcal{P}_0$.



(a) (b) (c)

Fig. 4. 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 $\overline{\mathbb{X}_0^{\Phi_2}}$ (light gray region) and some trajectories of the closed loop system (the initial states are marked by circles); (c) The maximal constrained control invariant set within $\mathbb{X}$.



(a) (b)

Fig. 5. Case study 2: (a) The set of satisfying initial states is shown in light gray; (b) Some satisfying trajectories of the closed loop system (the initial states are marked by circles).

B. Case Study 2 : Triple Integrator

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

$$A = \begin{bmatrix} 1 & 1 & 0.5 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 0.167 \\ 0.5 \\ 1 \end{bmatrix} \tag{25}$$

and $\mathbb{U} = \{u \in \mathbb{R} \mid -2 \leq u \leq 2\}$.

The specification is to reach a target region $\mathbb{T}$, while always staying inside a safe set $\mathbb{X}$ and avoiding obstacles $\mathbb{O}_1$ and $\mathbb{O}_2$, where $\mathbb{T} = \{x \mid -0.5 \leq x_i \leq 0.5, i = 1,2,3\}$, $\mathbb{X} = \{x \mid -4 \leq x_i \leq 4, i = 1,2,3\}$, $\mathbb{O}_1 = \{x \mid 0.5 \leq x_1 \leq 4, -4 \leq x_2 \leq 4, -4 \leq x_3 \leq 0.5,\}$, and $\mathbb{O}_2 = \{x \mid -4 \leq x_1 \leq -0.5, -4 \leq x_2 \leq 4, 2 \leq x_3 \leq 4\}$. These regions, which are all boxes, are shown in Fig. 5(b). Each box is represented using six predicates, one for each facet, where $[c_i^\top, c_{i+1}^\top, c_{i+2}^\top] = -I_3$, $i = 0, 6$; $[c_i^\top, c_{i+1}^\top, c_{i+2}^\top] = I_3$, $i = 3, 9$; $d_i = 4, i = 0, \ldots, 5$; $d_i = 0.5i = 6, \ldots, 11$ and $c_{12} = -e_3$, $d_{12} = 2$. The specification can be formally stated as the following scLTL formula:

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

The computation summary for this case study is given in the second row of Table I. $\overline{\mathbb{X}_0^{\Phi_3}}$ and some satisfying trajectories are shown in Fig. 5. Note that $\overline{\mathbb{X}_0^{\Phi_3}}$ covers 37% of the obstacle free safe region and any trajectory originating in $\overline{\mathbb{X}_0^{\Phi_3}}$ satisfies the specification in less than 10 steps.

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 all of the encountered controller synthesis problems of the type Prob. V.2 were infeasible and the states were partitioned until all of them became a beacon for a transition.

C. Case Study 3 : Toggle Switch

In this case study, we apply the proposed methods to a PWA system, which models a synthetic biological repressor network known as the toggle switch [33]. The system and the specification are adapted from [16], where the goal was to find a constant control for each operating region from LTL specifications. The state space of the PWA system is partitioned into 36 polytopic operating regions that are shown in Fig. 6. We assume that the controls are constrained to set $\mathbb{U} = \{u \in \mathbb{R}^2 \mid -2.5 \leq u_i \leq 2.5, i = 1,2\}$ and we omit the dynamics due to the space constraints.

The specification is to reach a target region $\mathbb{T}$, while staying inside a safe set $\mathbb{X}$ and avoiding a region $\mathbb{O}$. The polytopic regions and linear predicates used to define these regions are shown in Fig. 6. The specification is formally stated as the following scLTL formula:

$\Phi_4 = (p_0 \wedge p_1 \wedge p_2 \wedge p_3 \wedge \neg(p_4 \wedge p_5 \wedge p_6 \wedge p_7))\mathcal{U}(p_0 \wedge p_8 \wedge p_3 \wedge p_9)$.

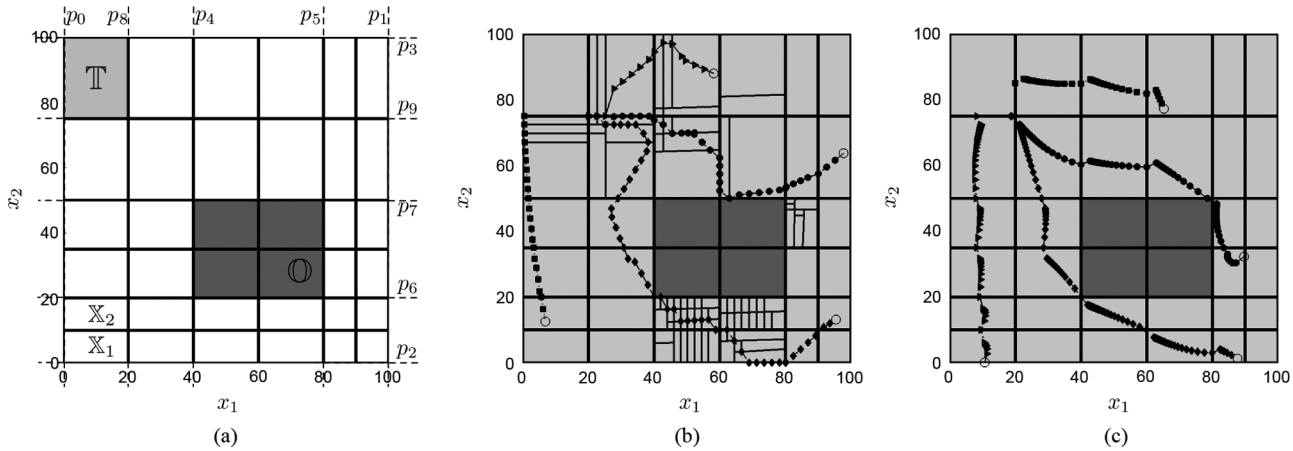


Fig. 6. Case study 3: (a) The regions and the corresponding linear predicates. The predicates are shown in the half planes where they are satisfied. The operating regions of the PWA system are shown with thick black borders and two of them are labeled. (b-c) The set of satisfying initial states $\overline{\mathbb{X}_0^{\Phi_4}}$ (light gray region) and some trajectories of the closed loop system (the initial states are marked by circles). (b) Vertex interpolation; (c) Contractive sets.

The computation summary for this case study is given in the third row of Table I. We apply both of the controller synthesis methods presented in Section V. When the vertex interpolation method is used with a maximal transition cost 5 ($N = 5$, see Remark V.3), the refinement algorithm terminates after 75 iterations. The resulting control strategy guarantees that any trajectory originating in $\overline{\mathbb{X}_0^{\Phi_4}}$, which covers the safe region $\mathbb{X}$ except the region $\mathbb{O}$, satisfies the specification within 54 steps. When the contractive sets method is employed, the dual automaton is not refined since all the states have finite costs. Hence, the computation time is much less as given in Table I. In this case any trajectory originating in $\overline{\mathbb{X}_0^{\Phi_4}}$ satisfies the specification within 175 steps. $\overline{\mathbb{X}_0^{\Phi_4}}$, the regions of automaton $\mathcal{A}^C$ states and some sample trajectories are shown in Fig. 6.

Remark VIII.1: The applicability and performance of the developed tool was assessed in comparison with existing tools. For example, Pessoa [34] and ConPAS2 [16] implement synthesis of control strategies for discrete time systems from reach-avoid and LTL specifications, respectively. These tools first construct a finite abstraction of the system, synthesize a control strategy for the abstract model and then refine this strategy to the original system.

Pessoa constructs a finite approximate simulation (or bisimulation) quotient of a dynamical system by quantizing the state and the control spaces, and supports controller synthesis from reach-avoid specifications. In contrast our approach does not assume quantized controls and it is iterative. As a result it is difficult to compare the two methods. However, the computation times are comparable for the double integrator dynamics, e.g., 109 s reported in [34] as compared to our first entry (89 s) in Table I, while the setting considered in this paper (control and state spaces, and scLTL specifications) is of significantly higher complexity.

ConPAS2 constructs a finite abstraction of a PWA system for a given partition of its state space through construction of equivalence classes in the control space. While the approach is robust with respect to control perturbations, and can handle any LTL specification, it is not iterative, and hence relies on the initial partition. The computational complexity associated with the construction of the abstraction is characterized by the number of polyhedral operations performed and it depends exponentially on the size of the initial partition. On the other hand, our approach is iterative and the complexity is defined with respect to the specification automaton. The number of the polyhedral operations performed and the LPs solved in the first iteration of our approach are linear in the number of transitions of the initial dual automaton. The tool ConPAS2 was tested on Case Studies 1 and 3. For Case Study 1, ConPAS2 did not find any satisfying initial states on the partition induced by the linear predicates, and for Case Study 3, although we used a smaller control set $\mathbb{U}$, the set of satisfying initial states found by ConPAS2 was smaller than $\overline{\mathbb{X}_0^{\Phi_4}}$. These are due to our refinement procedure and control strategies (in [16], a constant control input is assigned to each partition set).

As the specification from Case Study 2 is simple enough to represent as a reach-avoid problem, the set of satisfying initial states can be found through iterative computation of the one step controllable sets within $\mathbb{X}_S := \mathbb{X} \setminus \{\mathbb{O}_1 \cup \mathbb{O}_2\}$, i.e., $\overline{\mathbb{X}_0^{\Phi_3}} = \cup_{i=0,\ldots,n} \mathbb{T}_i$, where $\mathbb{T}_i = Pre(\mathbb{T}_{i-1}) \cap \mathbb{X}_S$ and $\mathbb{T}_0 = \mathbb{T}$. This computation took 556 minutes. The simple case of finding the set of states $\mathbb{X}_0^N \subseteq \mathbb{X}_S$ that can reach $\mathbb{T}$ in N steps can be directly handled by the standard constrained synthesis methods, such as predictive control. Using the *mpt_ownMPC* function implemented in the MPT toolbox, $\cup_{N=1,\ldots,5} \mathbb{X}_0^N$ (which covers 93% of $\overline{\mathbb{X}_0^{\Phi_3}}$) were computed in 11 min. For $N = 6$, no solution was returned within a day. These findings indicate the utility of the developed tool for obstacle-avoidance problems, compared with existing MPC-type solutions.

IX. Conclusion

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 set contraction 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 several non-trivial case studies.

Directions for future research include the formulation of less conservative sufficient conditions for feasibility of Prob. V.2 and the synthesis of provably-correct (i.e., satisfying temporal logic constraints) optimal control strategies.

REFERENCES

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

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

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

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

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

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

[7] G. Batt, D. Ropers, H. deJong, 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 *Proc. 13th Int. Conf. Intell. Syst. Molecular Biol.*, 2005, pp. 19–28.

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

[9] G. Batt, C. Belta, and R. Weiss, "Temporal logic analysis of gene networks under parameter uncertainty," *IEEE Trans. Circuits Syst. & IEEE Trans. Autom. Control, Joint Special Issue Syst. Biol.*, 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 *Proc. IEEE Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 3116–3121.

[11] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ -calculus specifications," in *Proc. IEEE Conf. Decision 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 *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, USA, 2004, pp. 4417–4422.

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

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

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

[16] B. Yordanov, J. Tumova, C. Belta, I. Cerna, and J. Barnat, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1491–1504, Jun. 2012.

[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 *Proc. IEEE Conf. Decision Control*, 2007, pp. 188–193.

[18] L. Habets and J. v. Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, pp. 21–35, 2004.

[19] M. Broucke, "Reach control on simplices by continuous state feedback," *SIAM J. Control Optim.*, vol. 48, no. 5, pp. 3482–3500, 2010.

[20] S. V. Rakovic and D. Mayne, "Robust model predictive control for obstacle avoidance: Discrete time case," *Lecture Notes Control Inform. Sci. (LNCIS)*, vol. 358, pp. 617–627, Sep. 2007.

[21] F. Blanchini, S. Miani, F. Pellegrino, and B. v. Arkel, "Enhancing controller performance for robot positioning in a constrained environment," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 5, pp. 1066–1074, 2008.

[22] E. A. Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for discrete-time linear systems," *Hybrid Syst.: Computat. Control. ACM*, pp. 95–104, 2012.

[23] G. M. Ziegler, *Lectures on Polytopes*. New York: Springer, 2007.

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

[25] T. Latvala, "Efficient model checking of safety properties," in *Proc. 10th Int. SPIN Workshop Model Checking Software.* , 2003, pp. 74–88.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT press, 2001.

[27] P. Tondel, T. A. Johansen, and A. Bemporad, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, no. 5, pp. 945–950, 2003.

[28] F. Blanchini and S. Miani, *Set-Theoretic Methods in Control*. Boston, MA, USA: Birkhauser, 2008.

[29] V. Spinu, N. Athanasopoulos, M. Lazar, and G. Bitsoris, "Stabilization of bilinear power converters by affine state feedback under input and state constraints," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 8, pp. 520–524, 2012.

[30] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Rob. Res.*, vol. 30, no. 14, pp. 1695–1708, 2011.

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

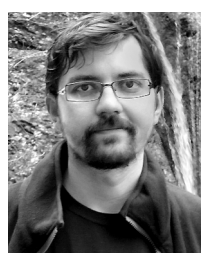
[32] I. G. Optimization, 2013 [Online]. Available: http://www.gurobi.com

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

[34] M. Mazo, A. Davitian, and P. Tabuada, PESSOA: A Tool for Embedded Controller Synthesis [Online]. Available: http://sites.google.com/a/cyphylab.ee.ucla.edu/pessoa/publications/Pessoa.pdf

Ebru Aydin Gol (S'10) received the B.S. degree in computer engineering from Orta Dogu TeknikÜniversitesi, Ankara, Turkey, in 2008, the M.S. degree in computer science from Ecole Polytechnique Fèdèrale de Lausanne, Lausanne, Switzerland, in 2010, and is currently pursuing the Ph.D. degree in systems engineering at Boston University, Boston, MA.

Her research interests include verification and control of dynamical systems, optimal control, and synthetic biology.



Mircea Lazar (M'06) was born in Iasi, Romania, in 1978. He received the M.Sc. degree in control engineering from the Technical University "Gh. Asachi" of Iasi, Romania, in 2002 and the Ph.D. degree in control engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2006.

Since 2006 he has been an Assistant Professor in the Control Systems group of the Electrical Engineering Faculty at the Eindhoven University of Technology. His research interests lie in stability theory, scalable Lyapunov methods and formal methods, and model predictive control.

Dr. Lazar received the EECI (European Embedded Control Institute) Ph.D. award.

Calin Belta (SM'11) is an Associate Professor in the Department of Mechanical Engineering, Department of Electrical and Computer Engineering, and the Division of Systems Engineering, Boston University, Boston, MA.

He is an Associate Editor for the *SIAM Journal on Control and Optimization* (SICON). His research focuses on dynamics and control theory, with particular emphasis on hybrid and cyber-physical systems, formal synthesis and verification, and applications in robotics and systems biology.

Dr. Belta received the Air Force Office of Scientific Research Young Investigator Award and the National Science Foundation CAREER Award.