Language-guided controller synthesis for linear systems

Aydin Gol, E. (Ebru); Lazar, M.; Belta, C.

Published in:
IEEE Transactions on Automatic Control

DOI:
10.1109/TAC.2013.2295664

Published: 01/01/2014

Document Version
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. Dec. 2018
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 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

EMPORAL 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 “((¬OUT) ∧ (¬IU(R1 ∨ R2)))” requires convergence to target region I through regions R1 or R2 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
number of linear programs. The second solution is based on con-
ttractive sets. While more conservative, it only requires solving a
single linear program. The existing approaches on solving poly-
tope-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 control-
ling a linear system from one polytope to another is defined as a
case reachability problem [18], [19]. Similar to the pro-
posed vertex interpolation method, to solve the facet reacha-
ability 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 in-
terpolation 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 obsta-
cles. Furthermore, it does not necessarily involve paths char-
acterized 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 poly-
tope-to-polytope control problem. Second, we show how the
proposed language-guided approach can be extended to piece-
wise-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 iter-
avive construction of the abstraction is presented in Section IV.
The LP-based algorithms for solving polytope-to-polytope con-
trary 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 illustra-
tive examples are shown in Section VIII. Conclusions are sum-
marized in Section IX.

II. NOTATION AND PRELIMINARIES

For a set S, int(S), Co(S), #S, and 2S stand for its interior,
convex hull, cardinality, and power set, respectively. For λ ∈ R
and S ⊆ Rn, let λS := {λx | x ∈ S}. We use Rn, Hn, Z, and Z+
for the sets of real numbers, non-negative reals, integers,
and non-negative integers. For m, n ∈ Z+, we use Rn
and Rm×n to denote the sets of column vectors and matrices
with n and m × n real entries. In, Rn stands for the n × n identity
matrix. For a matrix A, A+, A-, A+, denote its r-th row and
j-th column, respectively. Given two arbitrary sets S, P ⊆ Rn,
S ⊕ P = {s + p | s ∈ S, p ∈ P} denotes their Minkowski sum.
A polyhedron (polyhedral set) in Rn is the intersection of a
finite number of open and/or closed half-spaces. A polytope is a
compact polyhedron. On a polytope P ∈ Rn, we use the set of vertices
of a polytope P. Both the V-representation (Co(V(P))) and the
H-representation (H := {x ∈ Rn | Hx ≤ h}), where matrix
H and vector hP have suitable dimensions) [23] of a polytope P
will be used throughout the paper.

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

Φ := p ν Φ ∨ Φ ∨ Φ Φ U Φ C Φ (1)

where p is an atomic proposition, (negation), ∨ (disjunction),
∧ (conjunction) are Boolean operators, and C (“next”), U (“until”),
and (“eventually”) are temporal operators.
An inﬁnite word w over a ﬁnite set Σ is an inﬁnite sequence
w = w0w1 ... , where wi ∈ Σ for all i ∈ Z+. Similarly, an inﬁnite word
w over a ﬁnite set Σ is a ﬁnite sequence w = w0 ... wd,
where d ∈ Z+ and wi ∈ Σ for all i = 0, ..., d. The semantics
of scLTL formulas is deﬁned over inﬁnite words over 2F as
follows:

Deﬁnition II.2: The satisfaction of an scLTL formula Φ at
position i ∈ Z+ of a word w over 2F, denoted by w |i Φ, is
recursively deﬁned as follows: 1) w1 |i = 0 i f p ∈ w, 2) w2 |i = w− i f p ∉ w1, 3) wi |i = 0 i f w1 = 0 i o r w1 = 0 i, 4) wi |i = 0 i f w1 = 0 i o r w1 = 0 i, 5) wi |i = 0 i f w1 = 0 i,
and 6) wi − i Φ if there is j > i such that wj = 0 i and w0 < ... < wk = 0 i.
A word w satisfying an scLTL formula Φ, written as w | − Φ,
where w | − Φ.

An important property of scLTL formulas is that, even though
they have inﬁnite-time semantics, their satisfaction is guaran-
teed in ﬁnite time. Explicitly, for any scLTL formula F over F,
any satisfying inﬁnite word over 2F contains a satisfying ﬁnite
preﬁx1. We use LΦ to denote the set of all (ﬁnite) preﬁxes of
all satisfying inﬁnite words.

Deﬁnition II.3: A deterministic inﬁnite state automaton (FSA)
over a set of atomic predicates, Σ is a tuple A = (Q, Σ, →, Q0, F),
where Q is a set of states, Σ is a set of symbols, and → is a transition
relation, Q0 ∈ Q is a set of initial states, and F ⊆ Q is a set of
ﬁnal states.
An accepting run rA of an automaton A on a ﬁnite word
w = w0 ... wd over Σ is a sequence of states ra = r0 ... rd+1
such that q0 ∈ Q0, qd+1 ∈ F and (q1, q2, q3+1) ∈ →A for all i = 0, ... , d.
The set of all words corresponding to all of the accepting runs
of A is called the language accepted by A and is denoted as LΦ.
For any scLTL formula Φ over F, there exists a FSA A with
input alphabet 2F that accepts the preﬁxes of all the satisfying
words, i.e., LΦ[24]. There are algorithmic procedures and off-
the-shelf tools, such as check2[25], for the construction of such
an automaton.

Deﬁnition II.4: Given a FSA A = (Q, Σ, →A, Q0, F), its
dual automaton is a tuple A∗ = (Q∗, →∗, τ ∗, Q∗0, F),
where Q∗ = {q, σ, q′}, (q, σ, q′) ∈ →A, τ ∗ = {(q, σ, q′) : (q, σ, q′) ∈ →A},
and F = {q, σ, q′, q′} ∈ F}. For i = 0, ... , d, τ ∗ : Q∗0 → τ ∗(q, σ, q′) = σ, Q∗0 = {q0, σ, q′} ∈ Q0, and F = {q, σ, q′, q′} ∈ F}.
Informally, the states of the dual automaton A∗ are the tra-
Golding of the automaton A. →∗ ⊆ Q∗0 × Q∗ is a transition
relation and a transition is deﬁned between two states of A∗ if

1We 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 A. The set of output symbols of \( A^D \) is the same as the set of symbols of A, i.e., \( \Sigma \), \( \tau^D \) is an output function. For a state of \( A^D \), \( \tau^D \) produces the symbol that enables the transition in A. The set of initial states \( Q_0^D \) of \( A^D \) is the set of all transitions that leave an initial state in A. Similarly, the set of final states \( F^D \) of \( A^D \) is the set of transitions that end in a final state of A.

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

**Proposition II.5**: The output language of the dual automaton \( A^D \) coincides with the language accepted by the automaton A, i.e., \( L_A = L_{A^D} \).

**Proof**: \( \Rightarrow \): For every word \( w = w_0 \ldots w_d \) accepted by automaton A, there exists a run \( r_A = q_0 \ldots q_{d+1} \) such that \( q_i \in Q_0 \), \( q_{d+1} \in F \) and \( (q_i, w_i, q_{i+1}) \in \tau \) for all \( i = 0, \ldots, d \). The transition sequence of this run corresponds to a run \( r_D^D \) of the dual automaton which generates the word \( w \) since \( (w, w_i, q_{i+1}) \in F^D \) and \( ((q_i, w_i, q_{i+1}), (q_i, w_{i+1}, q_{i+1})) \in \tau^D \) for all \( i = 0, \ldots, d \).

Similarly, a run \( r_D = (q_0, w_0, q_1), \ldots, (q_d, w_d, q_{d+1}) \) of \( A \) that produces the word \( w = w_0 \ldots w_d \) yields a run \( r_A = q_0 \ldots q_{d+1} \) of A which accepts the word \( w \).

**III. PROBLEM FORMULATION**

Consider a discrete-time linear control system of the form\(^2\)

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

where \( A \in \mathbb{R}^{n \times n} \) and \( H \in \mathbb{R}^{n \times m} \) 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 \( P = \{ p_i \}_{i=0}^\infty \) for some \( \ell \geq 1 \) be a set of atomic propositions given as linear inequalities in \( \mathbb{R}^n \). Each atomic proposition \( p_i \) induces a half-space

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

A trajectory \( x_0, x_1, \ldots \) of system (2) produces a word \( P_0P_1 \ldots \) where \( P_i \subset P \) is the set of atomic propositions satisfied by \( x_i \), i.e., \( P_i = \{ p_j \mid x_j \in [p_i] \} \). 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 \( \Phi \) over \( P \), we use \( \overline{\Phi}^D \) to denote the set of all initial states of system (2) for which there exist control sequences producing trajectories satisfying \( \Phi \).

**Problem III.1**: Given an scLTL formula \( \Phi \) over a set of linear predicates \( P \) and a dynamical system as defined in (2), construct a set of initial states \( X^D_0 \subset \overline{\Phi}^D \) (possibly \( \Phi^D \)) and a feedback control strategy such that all the closed-loop trajectories originating in \( X^D_0 \) satisfy \( \Phi \).

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 \( \Phi \). 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 \( \Phi \) are accepted by a FSA \( A = (Q, \Sigma, \tau_0, F) \). The dual automaton \( A^D = (Q^D, \tau^D, 2^P, \overline{\Phi}^D) \) is constructed by intersecting the states and the transitions of the automaton A (Def. II.4). As the transitions of A become states of \( A^D \), elements from \( 2^P \) label the states and define polyhedral sets within the state-space of system (2).

1) **Automaton Representation**: A FSA \( A \) with input alphabet \( 2^P \) that accepts the language of an scLTL formula \( \Phi \) over \( P \) is constructed with the tool scheck2 [25]. This tool labels each transition of the produced FSA with a disjunctive normal form (DNF) \( C_1 \lor C_2 \lor \cdots \lor C_d \), 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 \( D = C_1 \lor C_2 \lor \cdots \lor C_d \), \( P_{C_i} := [p_i] \cap \cdots \cap p_i \) denotes the set of states of system (2) that satisfy \( C_i = p_1 \land \cdots \land p_k \), where \( i \in \{0, \ldots, i\} \), \( i_j \in \{0, \ldots, k\} \) for all \( j \in c \), and \( P_D := \bigcup_{i=1}^d P_{C_i} \) 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 polyhedral. Before constructing the dual automaton, each DNF formula \( C_1 \lor C_2 \lor \cdots \lor C_d \) is simplified by applying the following rules:

1) **Empty set elimination**: \( C_i \) is eliminated if the corresponding region is empty, i.e., \( P_{C_i} = \emptyset \). 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., \( P_{C_i} \subseteq P_{C_j} \). 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.
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 $P_\Phi \subseteq X$ 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 \land p_1 \land p_2 \land [u(p_1 \land p_2 \land p_3 \land p_4) \land (p_1 \land p_2 \land p_3 \land p_4) \land (p_1 \land p_2 \land p_3 \land p_4) \land (p_1 \land p_2 \land p_3 \land p_4) \land (p_1 \land p_2 \land p_3 \land p_4)] \}

over $P = \{p_0, p_1, p_2, p_3, p_4\}$, where $c_0 = [-1, 1]^T$, $d_0 = 0$, $c_1 = [-1, 1]^T$, $d_1 = 4$, $c_2 = [0, 1]^T$, $d_2 = -0.1$, $c_3 = [-1, 0]^T$, $d_3 = -3$, $c_4 = [1, 0]^T$, $d_4 = 5$. The trajectories that satisfy $\Phi_1$ evolve in the region $p_0 \cap [p_1 \land p_2]$ until they reach the target region $[p_1 \land p_2 \land p_3 \land p_4]$. The regions defined by this set of predicates and the compact representation of a FSA that accepts the language satisfying formula $\Phi_1$ 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 $p_4 \land p_3 \land p_2 \land p_1$, corresponds to two transitions labeled by $p_4 \land p_3 \land p_2 \land p_1$ and $p_4 \land p_3 \land p_2 \land p_1$, 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 $x^D$ in that state. The simplification deletes $(-p_4 \land p_3 \land p_2 \land p_1)$ from the self transition of the state labeled with “$0$” to the state labeled with “$1$”, which is labeled by $p_4 \land p_3 \land p_2 \land p_1$.

2) Initial Pruning: The feasibility of the transitions of the dual automaton is first checked by considering the particular dynamics of system (2) and the set $P$ where the control input takes values. $Post(P)$ denotes the set of states that can be reached from $P$ in one discrete-time instant under the dynamics (2) and $Post(P)$ is formally defined as

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

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

$$Post(P) = \text{Co}(\{Ax + Bu \mid (x, u) \in V(P) \times V(U)\}). \quad (6)$$

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 \( A^D = (Q^D, \rightarrow^D, \tau^D, Q_0^D, F^D) \).

**Output:** Pruned dual automaton \( A^{D_0} = (Q^{D_0}, \rightarrow^{D_0}, \tau^{D_0}, Q_0^{D_0}, F^{D_0}) \).

1: \( Q^{D_0} := Q^D \)
2: \( \rightarrow^{D_0} := \rightarrow^D \cup \{(q, q') | P_{\text{Post}}(\mathcal{P}_q) \cap \mathcal{P}_{q'} = \emptyset\} \)
3: while \( Q^{D_0} \neq \emptyset \) do
4: for all \( q \in Q^{D_0} \) do
5: if \((q \notin F^D \text{ AND } \{q' \mid (q, q') \in \rightarrow^D \} = \emptyset\) \text{ OR } \((q \notin Q_0^D \text{ AND } \{q' \mid (q', q) \in \rightarrow^{D_0} \} = \emptyset\)) then
6: \( Q^{D_0} := Q^{D_0} \setminus \{q\} \)
7: \( Q^D := Q^D \cup \{q' \mid (q', q) \in \rightarrow^{D_0} \} \cup \{q' \mid (q', q') \in \rightarrow^{D_0} \} \)
8: \( \rightarrow^{D_0} := \rightarrow^{D_0} \cup \{(q, q') \mid (q, q') \in \rightarrow^{D_0} \} \cup \{(q', q') \mid (q', q') \in \rightarrow^{D_0} \} \)
9: end if
10: end for
11: end while
12: \( Q_{\text{ref}}^{D_0} = Q^D \cap Q^{D_0} \)

**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 \( A^{D_0} = (Q^{D_0}, \rightarrow^{D_0}, \tau^{D_0}, Q_0^{D_0}, F^{D_0}) \).

**Output:** Refined dual automaton \( A^{D_R} = (Q^{D_R}, \rightarrow^{D_R}, \tau^{D_R}, Q_0^{D_R}, F^{D_R}) \).

1: \( J((q, q')) := \text{FeasibilityTest}(q, q') \), for each \((q, q') \in \rightarrow^{D_0} \)
2: \( J^P := \text{ShortestPath}(J, F^{D_0}) \)
3: \( \text{CandidateSet} = \{(q_i, q_j) \mid (q_i, q_i) \in \rightarrow^{D_0}, J^P(q_i) = \infty, J^P(q_j) \neq \infty\} \)
4: \( k := 0 \)
5: while \( \text{CandidateSet} \neq \emptyset \) do
6: \( (q_k, q_{k+1}) := \min_{(q_i, q_j) \in \text{CandidateSet}} \{(q_i, q_j) \mid (q_i, q_j) \in \text{CandidateSet}\} \)
7: \( A^{D_{k+1}} := \text{Partitioning}(A^{D_k}, q_k) \)
8: \( J((q, q')) := \text{FeasibilityTest}(q, q') \), for each \((q, q') \in \rightarrow^{D_{k+1}} \)
9: \( J^P := \text{ShortestPath}(J, F^{D_{k+1}}) \)
10: \( \text{CandidateSet} := \{(q_i, q_j) \mid (q_i, q_j) \in \rightarrow^{D_{k+1}}, J^P(q_i) = \infty, J^P(q_j) \neq \infty\} \)
11: \( k := k + 1 \)
12: 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 \( B_{qq'} \) to denote the controller corresponding to transition \((q, q')\), which can be obtained as \( B_{qq'} := \mathcal{P}_q \cap \text{Pre}(\mathcal{P}_{q'}) \), where

\[
\text{Pre}(\mathcal{P}) = \{x \in \mathbb{X} \mid \exists u \in \mathbb{U} : Ax + Bu \in \mathcal{P}\}, \quad \forall \mathcal{P} \subseteq \mathbb{R}^n. \tag{7}
\]

If \( \mathcal{P} \) and \( \mathcal{U} \) are polytopes, then \( \text{Pre}(\mathcal{P}) \) 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 \( A^D \) using Dijkstra’s algorithm [26]. The Partitioning procedure, which will be presented in detail in the next subsection, partitions a state region and modifies \( A^D \) 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 \( \mathcal{P}_q \), i.e., \( B_{qq'} \), 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-\( B_{qq'} \) problem, since a trajectory originating in \( \mathcal{P}_q \) will reach \( \mathcal{P}_{q'} \) without leaving \( \mathcal{P}_q \) only through the beacon \( B_{qq'} \). 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 \( \mathcal{P}_q = B_{qq'} \), since \( B_{qq'} \) 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 $\mathcal{J}^i(q_i) = \infty$ and a transition $(\{q_i, q_j\})$ to a state that has a finite cost $\mathcal{J}^j(q_j) < \infty$ is chosen as a candidate state for partitioning (lines 3 and 10). Then, a state $q_i$ 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 $q \in \mathcal{Q}_D^0$ to preserve the automaton language. The partitioning procedure is summarized in Alg. 3.

**Algorithm 3 Partitioning**

**Input:** Dual automaton $\mathcal{A}^D = (\mathcal{Q}^D, \rightarrow^D, \mathcal{P}^D, \tau^D, \mathcal{Q}^F_D, \mathcal{F}^D)$, a state $q_i \in \mathcal{Q}_D^0$.

**Output:** Partitioned automaton $\mathcal{A}^{D'} = (\mathcal{Q}^{D'}, \rightarrow^{D'}, \mathcal{P}^{D'}, \tau^{D'}, \mathcal{Q}^{F'}_{D'}, \mathcal{F}^{D'})$, $\mathcal{P}^{D'} = \{q_1, \ldots, q_d\} \subset \mathcal{P}_{q_i}$, and $\mathcal{P}_q \cap \mathcal{P}_{q_i} = \emptyset$ if $i \neq j$.

1: $\mathcal{Q}^{D'} = \{q_1, \ldots, q_d\}$, $\mathcal{P}_{q_i} = \cup_{1 \leq i \leq d} \mathcal{P}_{q_i}$, and $\mathcal{P}_q \cap \mathcal{P}_{q_i} = \emptyset$ if $i \neq j$.

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

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

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

5: $\mathcal{Q}^{F'} := \mathcal{Q}^F_D \setminus \{\mathcal{P}^{D'}\}$.

6: $\mathcal{Q}^{D'} = \mathcal{Q}^D \cup \mathcal{Q}^{D'}$ if $\mathcal{Q}^{D'} \not\subset \mathcal{Q}^D$.

7: $\rightarrow^{D'} := \rightarrow^D \cup \{(q, q_i) \mid (q, q_i) \in \rightarrow^D, q_i \in \mathcal{Q}^P \}$.

8: $\mathcal{P}_q \cap \mathcal{P}_{q_i} = \emptyset$.

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 envelope. The hyperplane, which maximizes the radius of the Chebyshev polytope and the intersection of two polytopes is a polytope.

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 $A = I_2$, $B = I_2$, $U = \{u \in \mathbb{R}^2 \mid 0 < u_1 < 0.2, -0.1 < u_2 < 0.2\}$ 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 $q_0$ 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 $\mathcal{B}_{q_2}$, generates the state regions and the automaton shown in Fig. 3(c) and (d). As $\text{Post}(\mathcal{P}_{q_1} \cap \mathcal{P}_{q_2}) = \emptyset$, the transition $(q_3, q_2)$ is not added. In the next iteration $q_1$ is partitioned using $\mathcal{B}_{q_1}$ 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 $\mathcal{X}_{0,1}^{\Phi}$ is non-empty. Given an arbitrary iteration $i \geq 1$ of Alg. 2, the set $\mathcal{X}_{0,1}^{\Phi}$ 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 $\mathcal{X}_{0,1}^{\Phi}$ satisfies the formula $\Phi$, and

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

Proof: (1) A finite path cost for a state $q_0 \in \mathcal{Q}^D_0$ 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 $J^D(q_0) = \sum_{j=0}^{d-1} J^D((q_j, q_{j+1}))$. 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 $\mathcal{P}_q$, 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 $\mathcal{L}_{\mathcal{A}^{D_0}} \subseteq \mathcal{L}_\Phi$. Consequently, $\mathcal{L}_{\mathcal{A}^{D_i}} \subseteq \mathcal{L}_\Phi$ and the resulting trajectories satisfy the formula.

(ii) For any $x \in \mathcal{X}_{0,i-1}^{\Phi}$, there exists an accepting automaton run $r_{D'} = q_0 \ldots q_d$ with $x \in \mathcal{P}_{q_0}$ and $J^D(q_0) = \sum_{j=0}^{d-1} J^D((q_j, q_{j+1})) < \infty$. Let $q_0$ be the state chosen for partitioning at iteration $i$. Then, $J^D((q_0, q_{j+1})) = \infty$ and $q_0 \neq q_j$ for all $j = 0, \ldots, d$ as $J^D(q_j) < \infty$ for all $j = 0, \ldots, d$. As only the transitions adjacent to $q_0$ are affected by partitioning, $q_0 \in \mathcal{Q}^D$ 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 $\mathcal{A}^{D_0}$ with finite cost.
and thus, \( x \in X_{0}^{\phi} \). Observing that \( x \in X_{0,4}^{\phi} \) was chosen arbitrarily completes the proof. 

The automaton refinement algorithms presented in this section generate a finite set of polytope-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 \( P_{q} \)-to-\( P_{q'} \) control problem. By the definition of \( B_{qq'} \), 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 \( x \in B_{qq'} \), that reaches \( P_{q'} \) in one discrete-time instant. The second problem concerns the construction of a control law which generates a closed-loop trajectory, for all \( x \in P_{q} \), that reaches \( B_{qq'} \) in a finite number of discrete-time instants. These synthesis problems are formally stated next.

**Problem V.1:** Let \( B \) and \( P \) be polytopes in \( \mathbb{R}^{n} \) with \( B \subseteq \text{Prel}(P) \) and consider system (2). Construct a state-feedback control law \( g : \mathbb{R}^{n} \rightarrow \mathbb{R}^{n} \) such that

\[
Ag + Bg(x) \in P, \quad g(x) \in U, \quad \forall x \in B.
\]

1) **Problem V.2:** Let \( N \in \mathbb{N} \), and \( N \geq 1 \), \( B \) and \( P \) be polytopes in \( \mathbb{R}^{n} \) with \( B \subseteq P \) and consider system (2). Construct a state-feedback control law \( g : \mathbb{R}^{n} \rightarrow \mathbb{R}^{n} \) such that for all \( x_{0} \in P \)

\[
x_{k+1} = Ax_{k} + Bg(x_{k}), \quad \forall k = 0, \ldots, N - 1, \\
x_{k} \in P, \quad g(x_{k}) \in U, \quad \forall k = 0, \ldots, N - 1, \\
x_{N} \in B.
\]

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

\[
g(x) := \sum_{i=1}^{\#V(B)} \lambda_{i}u^{i}
\]

where \( \lambda_{i} \in \mathbb{R}, 0 \leq \lambda_{i} \leq 1 \), are such that \( x = \sum_{i=1}^{\#V(B)} \lambda_{i}u^{i} \), solves Prob. V.1. The evaluation of the control law (10) requires on-line calculation of the coefficients \( \{\lambda_{i}\}_{i=1,\ldots,\#V(B)} \), 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 \( B \). 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 \( \{\lambda_{i}\}_{i=1,\ldots,\#V(B)} \) 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 \( B \), 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 \( \{v^{i}\}_{i=1,\ldots,\#V(P)} \) be the vertices of \( P \) and let \( \{u^{i}\}_{i=1,\ldots,\#V(P)} \) denote a corresponding set of finite sequences of control actions, where \( u^{i} := \{u_{k}^{i}\}_{k=0,\ldots,N-1} \) for all \( i = 1, \ldots, \#V(P) \) and \( N \geq 1 \). For each \( v^{i} \in V(P) \) define the following set of linear equality and inequality constraints in the variables \( \{u^{i}\}_{i=1,\ldots,\#V(P)} \):

\[
x_{k+1} = Ax_{k} + Bu_{k}, \quad \forall k = 0, \ldots, N - 1, \\
H_{P}x_{k} \leq h_{P}, \quad H_{U}u_{k} \leq h_{U}, \quad \forall k = 0, \ldots, N - 1, \\
H_{G}u_{N} \leq h_{G}.
\]

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 \( i = 1, \ldots, \#V(P) \), 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

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

(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 a closed-loop trajectory will reach $\mathcal{B}$ in at most $N^* - j^*$ discrete-time instants.

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 $g(x_k)$, both for its standard and faster variants presented above.

Remark V.3: In general, existence of a finite, common $\mathcal{E}$ 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

$$\mathcal{E}_\mathcal{B} := \{x^* \in \text{int}(\mathcal{B}) : \exists u^* \in U : x^* = Ax^* + Bu^*\}.$$
holds. Equation (16) and(19) imply that $\mathcal{M}(x_k) \leq \alpha_k^*$ and hence, $x_k \in B$. Furthermore, reordering the terms and taking the logarithm of both sides in Equation(19) yields

$$k \leq \frac{\ln \left( \frac{\alpha_k^* - \sigma_k}{\mathcal{M}(x_k) - \alpha_k^*} \right)}{\ln(\rho)}. \quad (20)$$

Noticing that $\mathcal{M}(x) \leq 1$ for all $x \in P$, the fact that $k \leq k^*$ follows directly from the definition of $k^*$ (Equation (17)). As the point $x_0 \in P$ 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 $x$ is an equilibrium point, then by augmenting the constraints of the LP (14) with the equilibrium point constraints, i.e., $(A + BK)x = 0$, and setting $\sigma_k$ 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 solves Prob. III.1 is composed of a finite state dual automaton $A^C$ (see Def. II.4) and a map $M$ from transitions of $A^C$ to state feedback controllers. The control strategy ($A^C, M$) is constructed as defined in Def. VI.1 from the dual automaton $A^D \cup$ and the cost functions $J$ and $J^F$ 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 $A^C$ has only finite cost transitions.

Definition VI.1 (Control Strategy): Given a dual automaton $A^D \cup = \langle Q^D \cup \rightarrow \rightarrow B^D \cup, 2^D \cup, \sigma^D \cup, Q^D \cup, F^D \cup, \rangle$, and cost functions $J : \rightarrow \rightarrow L^D \cup$ and $J^F : Q^D \cup \rightarrow \rightarrow L^F \cup$; the control strategy is a pair $\langle A^C, M \rangle$, where $A^C = \{ Q^C, \rightarrow \rightarrow C, 2^C, \sigma^C, Q^C, F^C, \}$ is a dual automaton defined as

$$Q^C = \{ q \in Q^D \cup | J^F(q) < \infty \},$$

$$\rightarrow \rightarrow C = \{ (q, g', q) | J((q, g')) < \infty, (q, g') \in \rightarrow \rightarrow D \cup, q' \in Q^C \},$$

$$\sigma^C : \rightarrow \rightarrow C \rightarrow \rightarrow 2^C,$$

$$Q^C = Q^C \cap \rightarrow \rightarrow C, F^C = F^D \cup$$

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

The union of the regions corresponding to the initial states of automaton $A^C$ defines the set of initial system states $X^S_0$, such that closed-loop trajectories originating in $X^S_0$ satisfy formula $\Phi$.

For a given accepting run $r_C = q_0 \ldots q_d$ of $A^C$, the time required to satisfy the specification for trajectories originating in $P_{q_0}$ is upper bounded by $\sum_{k=0}^{d-1} J((q_k, q_{k+1}))$ when $M_{r_C} = M((q_0, q_1)), \ldots, M((q_{d-1}, q_d))$ is applied. If the control sequences are chosen according to shortest paths for each $q_0 \in Q^C_0$, the time required to satisfy the specification starting from any state $x_0 \in X^S_0$ of system (2) is upper bounded by $\max_{q_0 \in Q^C_0} J^F(q_0)$.

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

Proof: The proof that all the trajectories of the closed loop system satisfy the formula follows immediately from Prop. IV.3 since $X^S_0 = X^S_{D \cup}$.

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 $w \in L_{\Phi}$ originates in $X^S_0$ i.e., $X^S_0 = \overline{\mathcal{F}}$.

Proof: To show that any satisfying trajectory originates in $X^S_0$, assume by contradiction that there exists $x_0 \notin X^S_0$ such that $x_0 \ldots x_d$ is a satisfying trajectory of system (2), i.e., $P_{x_0} \ldots P_{x_d} \subset L_{\Phi}$. Then by Prop. II.5, there exists an accepting run $r_D = q_0 \ldots q_d$ of the initial dual automaton $A^D \cup$ such that $x_k \in P_{q_k}$ for all $k = 0, \ldots, d$. The run $r_D$ induces a unique refined dual automaton run $r_{\Phi} = q_0^d \ldots q_d^k$ where $q_k^d$ and $q_k$ coincide or $q_k^d$ is obtained from $q_k$ through partitioning and $x_k \in P_{r_{\Phi}^k} \subseteq P_{q_k}$ for all $k = 0, \ldots, d$.

Let $r_{\Phi} = q_0^d \ldots q_d^k$ be obtained by eliminating consecutive duplicates in $r_D$. Then, for each $i = 0, \ldots, d - 1, s_i < s_{i+1}$ and $x_k \in P_{s_i}$ for all $k = s_i, s_i + 1, \ldots, s_{i+1} - 1$. Then, $x_0 \notin X^S_0$ indicates that $J^F(q_0) = \infty$. Hence, either $Post(P_{s_i}) \cap P_{s_{i+1}} = \emptyset$ or $J((q_{s_i}, q_{s_{i+1}})) = \infty$ for some $i = 0, \ldots, d$. Let $s_i$ be the maximal index where $Post(P_{s_i}) \cap P_{s_{i+1}} = \emptyset$ or $J((q_{s_i}, q_{s_{i+1}})) = \infty$. Therefore, $J((q_{s_i}, q_{s_{i+1}})) < \infty, \forall k = i + 1, \ldots, d - 1$ and $J^F(q_{s_i}) < \infty, \forall k = i + 1, \ldots, d$. As Alg. 2 terminates, it holds that

$$Post(P_{s_i}) \cap \{ P_{s_i} \cap \bigcup_{J^F(q) < \infty, (q, q') \in \rightarrow \rightarrow R} \} \neq \emptyset \quad (22)$$

for all $q$ with $J^F(q) = \infty$. Consequently, $Post(P_{s_i}) \cap P_{s_{i+1}} = \emptyset$. As $x_{s_{i+1}} \in P_{s_{i+1}}$ and $x_{s_{i+1}} \in P_{s_{i+1}}$, there is no control $u \in U$ that satisfies $x_{s_{i+1}} = Ax_{s_{i+1}} + Bu$. Therefore $x_0 \ldots x_d$ 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.,
is de for all of transitions are polytopes, and an admis-
imation (2), which depends on the control system (2)
in the partitions are poly-
. Note that, at each iteration
, and state procedures run is
, if and Dijkstra’s
is the number of modes (different dynamics),
a nd
has be the maximum number of transi-
tions. 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
polyhedral operations [see (6)]. Then, the feasibility of the states
are checked by traversing the underlying graph, which is poly-
nomial in
.

The complexity of Alg. 2 is dictated by the number of
iterations
, which depends on the control system (2),
the specification, and the employed polytope-to-polytope
troller synthesis method. Initially, transition
and
(costs are computed via
 and Dijkstra’s
algorithm, respectively. The run time of Dijkstra’s algorithm
is
. FeasibilityCheck procedure involves basic
polyhedral operations (see (7)) and controller synthesis, i.e.,
solving Prob. V.1 and Prob. V.2. The proposed controller syn-
thods are based on linear programming. The number
of constraints of the vertex interpolation-based solutions de-
pend 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
 be the maximum number of transitions
incident to a state
 for all iterations
. Then, the total number of FeasibilityCheck procedures run is
upper bounded by
. Note that, at each iteration
it is sufficient to compute costs
 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 spec-
ifications. A quick examination of the algorithms shows that
the method works if the regions
 in the partitions are poly-
topes, their pre-images
 through the system dynamics
are computable, the beacons
 of transitions are polytopes, and
the polytope-to-polytope controllers can be computed. Discrete-
time PWA systems with polytopic partitions satisfy these con-
ditions. In Section VII-A, we discuss the details of this exten-
sion. 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

\[ x_{k+1} = A_i x_k + B_i u_k + c_i, x_k \in \mathbb{X}_i, u_k \in \mathbb{U}_i, \quad i = 1, \ldots, l \]  

(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 \( A_i, B_i, c_i \) 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 an associated system dynamics \( (A_i, B_i, c_i) \) and an admi-
sible set of controls \( \mathbb{U}_i \) that can be used to compute the beacons
of transitions leaving the state \( q \) and Post(\( \mathbb{P}_q \)), 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.

IV. COMPATIBILITY WITH LINEAR TIME-TEMPORAL LOGIC

In this section, we extend the method to PWA systems,
for which we assume the dual automaton can be
computed. We discuss an extension of the method
from full LTL to PTL, which is not discussed in
this paper.

Algorithm 4 PWA Refinement

Input: Dual automaton \( \mathcal{A}^D = (Q^D, \rightarrow^D; 2^P, \tau^D, Q^D_0, F^D) \),
and regions \( \mathbb{X} = \bigcup_{i=1,\ldots,l} \mathbb{X}_i \)

Output: Refined automaton \( \mathcal{A}^D' = (Q^{D'}, \rightarrow^{D'}; 2^{P'}, \tau^{D'}, Q^{D'}_0, F^{D'}) \)

1: for all \( q \in Q^D \) do
2: \( q_D := \{ q_j | \mathcal{P}_{q_j} = \mathcal{P}_q \cap \mathbb{X}_j, \mathcal{P}_{q_j} \neq \emptyset \} \)
3: \( Q^D_{D'} := Q^D \cup q_D \)
4: \( \rightarrow^D := \rightarrow^D \cup \{ \bigcup_{\langle q, q' \rangle \in \rightarrow^D} \{ (q, q') | q_j \in q_D \} \} \)
\cup \{ \bigcup_{\langle q, q' \rangle \in \rightarrow^D} \{ (q', q_j) | q_j \in q_D \} \} \)
5: \( Q^{D'}_0 := Q^D_0 \cup q_D \) if \( q \in Q^D_0 \)
6: \( F^{D'} := F^D \cup \mathcal{P}_{q_D} \) if \( q \in F^D \)
7: \( \tau^{D'}(q_j) = \tau^D(q) \) for all \( q_j \in q_D \)
8: end for
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

\[
A = \begin{bmatrix}
1 & 1 \\
0 & 1
\end{bmatrix}, \quad B = \begin{bmatrix}
0.5 \\
1
\end{bmatrix}.
\]

We assume that the control constraint set is \( U = \{ u \mid -2 < u \leq 2 \} \). The specification is to visit region \( A \) or region \( B \), and then the target region \( T \), while always avoiding obstacles \( O_1 \) and \( O_2 \), and staying inside a safe region given by \( X = \{ x \mid -10 < x_1 < 1.55, -10 < x_2 < 2 \} \). The sets \( X, U \) and the obstacles \( O_1 \) and \( O_2 \) 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 \( x_0 \in X \) 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:

\[
\Phi_2 = ((P_0 \land p_1 \land p_2 \land p_3 \land \neg(p4 \land p_5) \land \neg(p_6 \land p_7)) \cup ((p_8 \land p_9 \land p_{10} \land p_{11}) \land \neg(p_6 \land p_7) \land \neg(p_8 \land p_9 \land p_{10} \land p_{11})) \cup ((p_5 \land p_7 \land p_{11} \land p_{12})).
\]

The computation summary for this case study is given in the first row of Table I. The set \( X_{0}^{\Phi_2} \) and a sample of satisfying trajectories of the closed loop system are shown in Fig. 4(b). Every trajectory originating in \( X_{0}^{\Phi_2} \) 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 \( X \). The method converged to the actual maximal set shown in Fig. 4(c) for the dynamics(24) and the given sets \( X \) and \( U \), 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 \( X \) starting from an invariant polytopic set \( P_0 \) which contains the origin, i.e., \( P_n = P rev(P_{n-1}) \cap X, 0 \in P_0 \).
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 $X_0^{\phi_1}$ (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 $X$.

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

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

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

The specification is to reach a target region $T$, while always staying inside a safe set $X$ and avoiding obstacles $O_1$ and $O_2$, where $T = \{ x \mid -0.5 \leq x_i \leq 0.5, i = 1, 2, 3 \}$, $X = \{ x \mid -4 \leq x_i \leq 4, i = 1, 2, 3 \}$, $O_1 = \{ x \mid 0.5 \leq x_3 \leq 4, -4 \leq x_2 \leq 4 \}$, and $O_2 = \{ x \mid -4 \leq x_1 \leq -0.5, -4 \leq x_2 \leq 4.2 \leq x_3 \leq 4 \}$. These regions, which are all boxes, are shown in Fig. 5(b). Each box is represented using six predicates, one for each facet, where $[c_i^1, c_{i+1}^1, c_{i+2}^1] = -I_3$, $i = 0, 6; [c_i^2, c_{i+1}^2, c_{i+2}^2] = I_3$, $i = 3, 9; d_i = 4, i = 0, \ldots, 5; d_i = 0.5i = 6, \ldots, 11$ and $c_{12} = -e_3, d_{12} = 2$. The specification can be formally stated as the following scLTL formula:

$$\Phi_3 = (p_0 \land p_1 \land p_2 \land p_3 \land p_4 \land p_5 \land \neg(p_3 \land \neg p_5 \land p_1 \land p_4 \land p_2 \land p_1) \land \neg(p_0 \land \neg p_6 \land p_1 \land p_4 \land p_1 \land p_2) \land \neg(p_5 \land p_7 \land p_8 \land p_9 \land p_{10} \land p_{11}))$$

The computation summary for this case study is given in the second row of Table I. $X_0^{\Phi_3}$ and some satisfying trajectories are shown in Fig. 5. Note that $X_0^{\Phi_3}$ covers 37% of the obstacle free safe region and any trajectory originating in $X_0^{\Phi_3}$ 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 $U = \{ u \in \mathbb{R}^2 \mid -2.5 \leq u_i \leq 2.5, i = 1, 2 \}$ and we omit the dynamics due to the space constraints.

The specification is to reach a target region $T$, while staying inside a safe set $X$ and avoiding a region $O$. 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:

$$\Phi_4 = (p_0 \land p_1 \land p_2 \land p_3 \land \neg(p_4 \land p_5 \land p_6 \land p_7) \land \neg(p_0 \land p_9 \land p_3 \land p_5)$$
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 $X^N_0$, which covers the safe region $X$ except the region $O$, 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 $X^N_0$ satisfies the specification within 175 steps. $X^N_0$, the regions of automaton $A^C$ 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 $U$, the set of satisfying initial states found by ConPAS2 was smaller than $X^N_{0}$.

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 $X_{\epsilon} := X \setminus \{O_1 \cup O_2\}$, i.e., $X^N_{0} = \bigcup_{1=1,\ldots,n} T_i$, where $T_i = P_{rv}(T_{i-1}) \cap X_S$ and $T_o = T$. This computation took 556 minutes. The simple case of finding the set of states $X^N_{0} \subseteq X_S$ that can reach $T$ in $N$ steps can be directly handled by the standard constrained synthesis methods, such as predictive control. Using the mpt_owmMPC function implemented in the MPT toolbox, $\cup_{N=1,\ldots,5} X^N_{0}$ (which covers 93% of $X^N_{0}$) 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 construction 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


Ebru Aydin Gol S‘(10) received the B.S. degree in computer engineering from Orta Dogu Teknik Üniversitesi, Ankara, Turkey, in 2008, the M.S. degree in computer science from Ecole Polytechnique Fédérale 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.