An Additive Cost Approach to Optimal Temporal Logic Control

Ebru Aydin Gol and Calin Belta

***Abstract*— This paper presents a provably-correct Model Predictive Control (MPC) scheme for a discrete-time linear system. The cost is a quadratic that penalizes the distance from desired state and control trajectories, which are only available over a finite horizon. Correctness is specified as a syntactically co-safe Linear Temporal Logic (scLTL) formula over a set of linear predicates in the states of the system. The proposed MPC controller solves a set of convex optimization problems guided by the specification. The objective of each optimization is to minimize the quadratic cost function and a distance to the satisfaction of the specification. The latter part of the objective and the constraints of the problem guarantee that the closed-loop trajectory satisfies the specification, while the former part is used to minimize the distance from the reference trajectories.**

I. Introduction

The goal in formal synthesis for dynamical systems is to compute control strategies from specifications expressed in formal languages, such as Linear Temporal Logic (LTL). As opposed to classical control specifications, such as stability and safety, one can easily express a complex specification, such as “Do not go to A unless B is visited before, eventually visit C and avoid D until C is visited”, as a temporal logic formula. Recent studies show that control strategies for dynamical systems can be generated from such specifications by adapting existing model checking and game-theoretic techniques [1]–[7].

In this work, as a natural extension to formal synthesis, we study the problem of synthesizing optimal control strategies from temporal logic specifications. In particular we consider specifications given in the form of syntactically co-safe linear temporal logic (scLTL) formulas. The syntactically co-safe fragment of LTL is rich enough to express a wide spectrum of finite-time properties of dynamical systems including the example given above. Despite the rich literature on formal synthesis for dynamical systems, the research on optimal formal synthesis is limited [3], [8].

We consider the following problem: given a discrete-time linear system, an initial system state, and an scLTL formula over linear predicates in the states of system, find a feedback control strategy such that the trajectory of the closed-loop system satisfies the formula and minimizes the cost. The cost is a quadratic function that penalizes the distance between the actual and desired state and control trajectories, which are only available over a finite horizon. Our approach consists of two main steps. The first step is the construction of an automaton from the specification formula and the system dynamics. The second step is the design of a Model Predictive Control (MPC) scheme over the automaton and system state spaces.

MPC has been shown to be an efficient and successful method in constrained control [13]. In the basic MPC setup, at each time step, the controller optimizes the cost over a finite horizon, finds the optimal control sequence, and applies the first control. The proposed MPC controller for solving optimal temporal logic control problem produces an optimal control sequence with respect to the available reference trajectory by solving a set of quadratic programs (QPs) guided by the specification.

MPC for dynamical systems from temporal logic specifications was first studied in [3], where a controller was derived for a finite abstraction of the system, and then refined to the original system. The satisfaction of the specification was guaranteed by assuming that the specification automaton had a known partial order structure. MPC of finite-state transition systems was studied in [9], where the satisfaction of the specification was guaranteed by a Lyapunov-type function. In [8], we extended this technique to dynamical systems with infinitely many states. Essentially, we used a Lyapunov-type function to implement a progress constraint in the optimization problem, which resembles terminal constraint in MPC. Here, we further extend this concept and define a contractive function, which we call a *potential* function. The value of this function decreases as a system trajectory makes progress towards a final automaton state. The objective of the optimization problem is a weighted sum of the potential and the quadratic cost. The weight of the potential increases at each time step with a user defined parameter γ, which guarantees that the trajectory eventually reaches a final automaton state. Moreover, by changing γ, we can enforce the trajectory to visit some regions (via the reference trajectory), before it reaches a final automaton state. Due to the progress constraint this was not possible in [8].

Due to space limitations, the results in this paper are stated without proofs. The proofs and additional details can be found in [10].

II. Notation and Preliminaries

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, respectively. 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.

This work was partially supported at Boston University by the NSF under grant CNS-1035588 and by the ONR under grants MURI 014-001-0303-5 and MURI N00014-10-10952.

Ebru Aydin Gol (ebru@bu.edu) and Calin Belta (cbelta@bu.edu) are with the Division of Systems Engineering at Boston University, Boston, MA, USA.

In this work, the control specifications are given as formulas of syntactically co-safe linear temporal logic (scLTL). A detailed description of the syntax and semantics of scLTL is beyond the scope of this paper and can be found in [11]. Roughly, an scLTL formula is built up from a set of atomic propositions P, standard Boolean operators : $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and temporal operators $\mathcal{X}$ (next), $\mathcal{U}$ (until) and $\mathcal{F}$ (eventually). The semantics of scLTL formulas are given over infinite words $\sigma = \sigma_0\sigma_1\dots$ where $\sigma_i \in 2^P$ for all i and 2^P is the power set of P. A word σ satisfies an scLTL formula ϕ, if it holds at the first position of the word σ. Informally, $\mathcal{X}\phi_1$ holds if ϕ_1 is true at the next position of the word, $\phi_1\mathcal{U}\phi_2$ holds if ϕ_2 eventually becomes true and ϕ_1 is true until this happens, and $\mathcal{F}\phi_1$ holds if ϕ_1 becomes true at some future position in the word.

While the semantics of scLTL formulas are defined over infinite words, their satisfaction is guaranteed in finite-time. Particularly, for any scLTL formula Φ over P, any satisfying infinite word over 2^P contains a finite *good* prefix and any word that contains a *good* prefix satisfies Φ. We use $\mathcal{L}_\Phi$ to denote the set of all (finite) good prefixes. We abuse the terminology and say that a finite word satisfies a formula if it contains a good prefix.

Definition II.1 A deterministic finite state automaton (FSA) is a tuple $\mathcal{A} = (Q, \Sigma, \rightarrow, Q_0, F)$, where Q is a finite set of states, Σ is a set of symbols, $\rightarrow \subseteq Q \times \Sigma \times Q$ 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 of an automaton $\mathcal{A}$ on a finite word $\sigma = \sigma_0 \dots \sigma_d$ over Σ is a sequence of states $r = q_0 \dots q_{d+1}$ such that $q_0 \in Q_0$, $q_{d+1} \in F$ and $(q_i, \sigma_i, q_{i+1}) \in\rightarrow$ for all $i = 0, \dots, d$. 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 Φ formula over P, there exists an FSA $\mathcal{A}$ with input alphabet 2^P that accepts the good prefixes of Φ, *i.e.* $\mathcal{L}_\Phi$ [11]. There are algorithmic procedures and tools, such as *scheck2* [12], for the construction of such an automaton.

Definition II.2 Given an FSA $\mathcal{A} = (Q, \Sigma, \rightarrow, Q_0, F)$, its dual automaton is a tuple $\mathcal{A}^D = (Q^D, \rightarrow^D, \Sigma, \tau^D, Q_0^D, F^D)$, where $Q^D = \{(q, \sigma, q') \mid (q, \sigma, q') \in\rightarrow\}$, $\rightarrow^D = \{((q, \sigma, q'), (q', \sigma', \bar{q})) \mid (q, \sigma, q'), (q', \sigma', \bar{q}) \in\rightarrow\}$, $\tau^D : Q^D \mapsto \Sigma$, $\tau^D((q, \sigma, q')) = \sigma$, $Q_0^D = \{(q, \sigma, q') \mid q \in Q_0\}$, and $F^D = \{(q, \sigma, q') \mid q' \in F\}$.

Informally, the states of the dual automaton $\mathcal{A}^D$ are the transitions of $\mathcal{A}$. There is a transition between two states of $\mathcal{A}^D$ if 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^D of a dual automaton is a sequence of states $r^D = q_0 \dots q_d$ such that $q_0 \in Q_0^D$, $q_d \in F^D$ and $(q_i, q_{i+1}) \in\rightarrow^D$ for all $i = 0, \dots, d-1$. An accepting run r^D produces a word $\sigma = \sigma_0 \dots \sigma_d$ over Σ such that $\tau(q_i) = \sigma_i$, 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 an FSA $\mathcal{A}$ guarantees that any word produced by $\mathcal{A}^D$ is accepted by $\mathcal{A}$, and any word accepted by $\mathcal{A}$ can be produced by $\mathcal{A}^D$, *i.e.* $\mathcal{L}_\mathcal{A} = \mathcal{L}_{\mathcal{A}^D}$.

III. Problem formulation

Consider a discrete-time linear control system of the form

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

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 the applied control at time $k \in \mathbb{Z}_+$, respectively. Let $x_0^r, x_1^r \dots$ and $u_0^r, u_1^r, \dots$ denote reference state and control trajectories, respectively. The stage cost at time k is defined with respect to x_k^r and u_k^r by $L : \mathbb{X} \times \mathbb{U} \rightarrow \mathbb{R}_+$:

$$L(x_k, u_k) = (x_k - x_k^r)^\top Q(x_k - x_k^r) + (u_k - u_k^r)^\top R(u_k - u_k^r), \tag{2}$$

where $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are positive definite matrices. We assume that, for some N, at time k the reference state and control trajectories of length N are known. At time k, the cost of a finite trajectory $x_k, \dots, x_{k+N-1}$ originating at x_k and generated by the control sequence $u_k, \dots, u_{k+N-1}$ is

$$\sum_{i=0}^{N-1} L(x_{k+i}, u_{k+i}). \tag{3}$$

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

$$[p_i] := \{x \in \mathbb{R}^n \mid c_i^\top x + d_i \leq 0\},\ c_i \in \mathbb{R}^n, d_i \in \mathbb{R}. \tag{4}$$

A trajectory $x_0, x_1, \dots$ of system (1) produces a word $P_0P_1\dots$ 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.

Problem III.1 Given an scLTL formula Φ over a set of linear predicates P, a dynamical system as defined in (1), and an initial state $x_0 \in \mathbb{X}$, find a feedback control strategy such that the closed-loop trajectory originating at x_0 satisfies Φ while minimizing the cost (3).

We propose a two-step solution to Problem III.1. In the first step, by using existing tools [5], we construct an automaton from the specification formula. The states of the automaton correspond to polyhedral subsets of the state space of system (1), and any satisfying trajectory of system (1) follows a sequence of polyhedral sets defined by an accepting run of the automaton. In the second step, we design an MPC controller that minimizes the cost over the available reference trajectory and the *distance* to a final automaton state, while ensuring that the resulting trajectory satisfies the specification. The constraints of the optimization problem ensure that the produced trajectory lies within an automaton path. A terminal cost function, which is a distance measure to a final automaton state, guarantees that the produced trajectory reaches a final automaton state, and hence it satisfies the specification while the cost over the available reference trajectory is minimized.

IV. Automaton Generation

A. Language-Guided Control

All words that satisfy the specification formula Φ over the set of linear predicates P are accepted by an FSA $\mathcal{A} = (Q, 2^P, \rightarrow, Q_0, F)$. The dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, \Sigma, \tau^D, Q_0^D, F^D)$ is constructed by interchanging the states and the transitions of $\mathcal{A}$ (Definition II.2). 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 (1). For a dual automaton state $q \in Q^D$, $\mathcal{P}_q \subset \mathbb{X}$ is used to denote the corresponding polyhedral set.

In [5], we developed a procedure for iterative refinement of the dual automaton and the corresponding polyhedral partition of the state space of system (1) with the goal of finding initial states and corresponding feedback control strategies producing satisfying trajectories. Starting with the initial dual automaton, at each iteration, we checked whether feedback controllers could be designed for the original system to "match" the transitions of the dual automaton. Essentially, each transition (q, q') induced a $\mathcal{P}_q - to - \mathcal{P}_{q'}$ control problem. This problem and the controller synthesis approach followed in this paper are presented below. At each iteration, each transition (q, q') was labeled with a cost $J((q, q'))$ that equaled the minimum number of discrete time steps necessary for all states in $\mathcal{P}_q$ to reach $\mathcal{P}_q'$ under the determined state feedback law. If no controller could be found, then the cost was set to infinity. The cost of a state q was 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 proposed in [5] iteratively partitions the regions with infinite cost, *i.e.* the regions for which there do not exist sequences of feedback controllers driving all the corresponding states to a region corresponding to a final state in the automaton. This procedure results in a monotonically increasing, with respect to set inclusion, set of initial states of system (1) for which an admissible control strategy can be found. As we showed in [5] (Theorem 6.1), if the refinement algorithm terminates, then all the satisfying trajectories of system (1) originate in the resulting set of initial states, denoted by $\mathbb{X}_0^\Phi$. In this paper, as our goal is to find a control strategy for a given initial state x_0, we terminate the algorithm at the ith iteration if $x_0 \in \mathbb{X}_{0,i}^\Phi$, where $\mathbb{X}_{0,i}^\Phi \subseteq \mathbb{X}$ is the union of the regions corresponding to start states of the automaton with finite path costs obtained at the ith iteration.

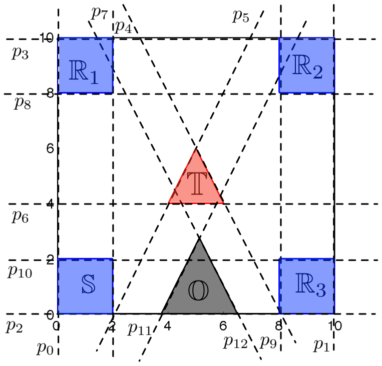


Fig. 1. The regions and the corresponding linear predicates for the specification from Example IV.2. The predicates are shown in the half planes where they are satisfied.

Transition controllers: To solve the $\mathcal{P}_q - to - \mathcal{P}_{q'}$ control problem induced by the transition (q, q') with $q \neq q'$ we first define the set $\mathcal{B}_{qq'} \subseteq \mathcal{P}_q$ as $\mathcal{B}_{qq'} := \{x \in \mathcal{P}_q | \exists u \in \mathbb{U} : Ax + Bu \in \mathcal{P}_{q'}\}$, and decompose the $\mathcal{P}_q - to - \mathcal{P}_{q'}$ control problem in two subproblems. The first problem, $\mathcal{B}_{qq'} - to - \mathcal{P}_{q'}$, consists of the computation of a control law that generates a closed-loop trajectory, for all $x \in \mathcal{B}_{qq'}$, which reaches $\mathcal{P}_{q'}$ in one discrete-time instant. The second problem, $\mathcal{P}_q - to - \mathcal{B}_{qq'}$ concerns the computation of a control law that generates a closed-loop trajectory, for all $x \in \mathcal{P}_q$, which reaches $\mathcal{B}_{qq'}$ in a finite number of discrete-time instants. By the definition of $\mathcal{B}_{qq'}$, the first problem is always feasible. If $(q, q') \notin \rightarrow^D$, then a control law solves the $\mathcal{P}_q - to - \mathcal{P}_{q'}$ control problem only if $\mathcal{P}_q \subseteq \mathcal{B}_{qq'}$. To solve the second problem, in [5] we presented two methods, one based on vertex interpolation and the other one on polyhedral Lyapunov functions. In this paper, we apply the polyhedral Lyapunov functions method:

Definition IV.1 For a transition $(q, q') \in \rightarrow^D$, suppose that the feedback control law $g : \mathcal{P}_q \rightarrow \mathbb{U}$ solves the $\mathcal{P}_q - to - \mathcal{B}_{qq'}$ control problem and is synthesized by using the polyhedral Lyapunov functions method. Then, there exist $x_{qq'} \in \mathcal{B}_{qq'}$ and $\rho_{qq'} \in [0, 1)$ such that

$$\mathcal{M}(Ax + Bg(x)) \leq \rho_{qq'} \mathcal{M}(x), \tag{5}$$

where

$$\mathcal{M}(x) := \max_{i=1,\ldots,w} W_{i\bullet}(x - x_{qq'}) \tag{6}$$

such that $W \in \mathbb{R}^{w \times n}$ and $\mathcal{P}_q = \{x \mid Wx \leq 1\}$. Then, the transition cost is defined as

$$\begin{aligned} J((q, q')) := 1 + \arg\min\{k \geq 0 \mid \\ \rho_{qq'}^k (\mathcal{P}_q \oplus \{-x_{qq'}\}) \subseteq (\mathcal{B}_{qq'} \oplus \{-x_{qq'}\})\}, \end{aligned} \tag{7}$$

where $\oplus$ is the Minkowski sum operator.

Example IV.2 Consider system (1) with $A = \left[\begin{smallmatrix} 0.99 & 0 \\ 0 & 0.98 \end{smallmatrix}\right]$, $B = I_2$, $\mathbb{U} = \{u \in \mathbb{R}^2 \mid -0.5 \leq u_i \leq 0.5, i = 1, 2\ \}$, $\mathbb{X} = \{x \in \mathbb{R}^2 \mid 0 \leq x_i \leq 10, i = 1, 2\ \}$ and $x_0 = \left[\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right]$. The regions of interest are defined using a set of linear predicates $P = \{p_0, \ldots, p_{12}\}$, which are shown in Figure 1. The specification is defined as "*A system trajectory originates in* $\mathbb{S}$*, eventually visits* $\mathbb{T}$*, and before visiting* $\mathbb{T}$ *it either visits* $\mathbb{R}_1$ *and* $\mathbb{R}_2$ *(in this order), or* $\mathbb{R}_3$*. Moreover, it does not visit* $\mathbb{O}$ *before it reaches* $\mathbb{T}$". The specification is translated to the following scLTL formula over P:

$\Phi_{ex} = ((p_7 \wedge p_{10}) \wedge (\mathcal{F}(p_4 \wedge \neg p_5 \wedge \neg p_6))) \wedge (\neg(\neg p_{11} \wedge p_{12})\mathcal{U}(p_4 \wedge \neg p_5 \wedge \neg p_6)) \wedge (\neg(p_4 \wedge \neg p_5 \wedge \neg p_6)\mathcal{U}((\neg p_8 \wedge \neg p_9) \vee (\neg p_9 \wedge p_{10}))) \wedge ((\neg(\neg p_8 \wedge \neg p_9)\mathcal{U}(p_4 \wedge \neg p_5 \wedge \neg p_6)) \vee (\neg(\neg p_8 \wedge \neg p_9)\mathcal{U}(p_7 \wedge \neg p_8)))$.

The refinement algorithm terminates at the first iteration with $\mathbb{X}_0^{\Phi_{ex}} = \mathbb{S}$, hence there exists a sequence of controllers such that all trajectories that originate from $\mathbb{S}$ and generated by these controllers satisfy the specification. The refined dual automaton has 101 states and 569 finite cost transitions.

In the remainder of the paper, for simplicity of notation, we use

$$\mathcal{A}^D = (Q^D, \rightarrow^D, \Sigma, \tau^D, Q_0^D, F^D) \tag{8}$$

to denote the (refined) dual automaton obtained at the last iteration of the algorithm presented above. We use $\mathcal{P}_q \subset \mathbb{X}$ to denote the polyhedral region of state $q \in Q^D$. We denote the transition cost function of $\mathcal{A}^D$ by $J : \rightarrow^D \longrightarrow \mathbb{Z}_+$.

Assumption IV.3 For any $q_0 \in Q^D$ there exists an automaton path $q_0 \ldots q_d$, $d \in \mathbb{Z}_+$ such that $J(q_i, q_{i+1}) < \infty$ for all $i = 0, \ldots, d-1$ and $q_d \in F^D$.

B. Potential Function

To enforce the satisfaction condition of a dual automaton, we define a real positive function that resembles a control Lyapunov function. In [9], such a function was used to enforce a Büchi acceptance condition on the trajectories of a finite deterministic transition system. In [8], we focussed on the acceptance condition of a finite state automaton and extended this concept to discrete time linear systems. Here, we further extend this idea and define a *contractive* function based on the transition controllers given in Definition IV.1. In addition to enforcing the accepting condition, this function allows us to steer the trajectory towards desired regions via the reference trajectory, which was not possible in [8].

Definition IV.4 A function $V : \bigcup_{q \in Q^D} \{\{q\} \times \mathcal{P}_q\} \rightarrow \mathbb{R}_+$ is called a potential function with contraction rate $\rho \in [0, 1)$ for a system (1) and a dual automaton (8) if it satisfies:

(i) $V(q, x) = 0$ for all $q \in F^D$.
(ii) For each $(q, x) \in \bigcup_{q \in Q^D} \{\{q\} \times \mathcal{P}_q\}$, it holds that if $V(q, x) \neq 0$ and $V(q, x) \neq \infty$, then there exists a control $u \in \mathbb{U}$ such that $x' = Ax + Bu$, $x' \in \mathcal{P}_{q'}$, $(q, q') \in \rightarrow^D$, and $V(q', x') \leq \rho V(q, x)$.

We define a potential function based on the transition cost function $J(\cdot)$. Informally, the potential function at (q, x), $q \in Q^D, x \in \mathcal{P}_q$ is defined as an upper bound for the time required to reach $\mathcal{P}_{q_f}$ from x by applying the polytope-to-polytope feedback controllers along a shortest path $qq_1 \ldots q_f$ from q to a final automaton state $q_f \in F^D$. In the rest of this section, we formalize this description and then show that this function satisfies the properties of Definition IV.4.

The set of all finite paths from q to F^D is denoted by $\mathbf{P}_q$:

$$\mathbf{P}_q = \{\mathbf{q} = q_0 q_1 \ldots q_d \mid d \in \mathbb{Z}_+, (q_i, q_{i+1}) \in \rightarrow^D, \\ i = 0, \ldots, d-1, q_d \in F^D, q_0 = q\}. \tag{9}$$

The cost $J^p(\mathbf{q})$ of an automaton path $\mathbf{q} = q_0 \ldots q_d$ is defined as the sum of the corresponding transition costs, *i.e.* $J^p(\mathbf{q}) = \sum_{i=0}^{d-1} J((q_i, q_{i-1}))$. The cost $J^s(q)$ of a state $q \in Q^D$ is the cost of the shortest path from q to a final state:

$$J^s(q) = \min_{\mathbf{q} \in \mathbf{P}_q} J^p(\mathbf{q}).$$

The successor $S(q)$ of a state $q \in Q^D$ is the state that succeeds q in the shortest path from q to F^D, *i.e.*

$$qS(q) \ldots = \arg\min_{\mathbf{q} \in \mathbf{P}_q} J^p(\mathbf{q}).$$

The continuous potential of a state $x \in \mathcal{P}_q$ with respect to a transition $(q, q') \in \rightarrow^D$ with $J((q, q')) \neq \infty$ is defined by the function $J^T : \bigcup_{(q,q') \in \rightarrow^D} \{\{(q, q')\} \times \mathcal{P}_q\} \longrightarrow \mathbb{Z}_+$ as

$$J^T((q, q'), x) = (J((q, q')) - 1)\mathcal{M}(x) + 1, \tag{10}$$

where $\mathcal{M}(\cdot)$ is defined as in (6).

Lemma IV.5 *For any* $(q, q') \in \rightarrow^D$ *with* $J((q, q')) \neq \infty$*, and* $x \in \mathcal{P}_q$*, the function* $J^T(\cdot, \cdot)$ *defined in* (10) *satisfies* $J^T((q, q'), x) \geq 1$ *and* $J^T((q, q'), x) \leq J((q, q'))$*.*

Finally, we define the potential function at (q, x) as

$$V(q, x) = \begin{cases} 0 & \text{if } q \in F^D, \\ J^T((q, S(q)), x) + J^s(S(q)) & \text{otherwise.} \end{cases} \tag{11}$$

Lemma IV.6 *For any* $q \in Q^D \setminus F^D$ *and* $x \in \mathcal{P}_q$*, the function* $V(\cdot, \cdot)$ *defined in* (11) *satisfies that*

$$J^s(S(q)) + 1 \leq V(q, x) \leq J^s(q).$$

Proposition IV.7 *According to Definition IV.4, the function defined in* (11) *is a potential function with contraction rate*

$$\rho = \max\{\max_{q \in Q^D} \frac{J^s(q)}{J^s(q) + 1}, \\ \max_{q \in Q^D \setminus F^D} \frac{(J((q, S(q))) - 1)\rho_{qS(q)}^{J(q,S(q))} + 1 + J^s(S(q))}{(J((q, S(q))) - 1)\rho_{qS(q)}^{J(q,S(q))-1} + 1 + J^s(S(q))}\}. \tag{12}$$

V. MPC Strategy

In Section IV, we outlined the generation of a dual automaton from the specification and the system dynamics (1), and defined a potential function. In this section, we design an MPC controller for a given dual automaton $\mathcal{A}^D = (Q^D, \rightarrow^D, \Sigma, \tau^D, Q_0^D, F^D)$ and a potential function $V : \bigcup_{q \in Q^D} \{\{q\} \times \mathcal{P}_q\} \rightarrow \mathbb{R}_+$. At each time step, the controller solves an optimization problem over $\bigcup_{q \in Q^D} \{\{q\} \times \mathcal{P}_q\}$.

Definition V.1 An automaton-enabled finite trajectory

$$\mathbf{T} = (q_0, x_0), \ldots, (q_N, x_N)$$

is a sequence of automaton (8) and system (1) state pairs such that

(i) for each $k = 0, \ldots, N-1$ there exists $u_k \in \mathbb{U}$ such that $x_{k+1} = Ax_k + Bu_k$,

(ii) $x_k \in \mathcal{P}_{q_k}$, for all $k = 0, \ldots, N$,

(iii) $(q_k, q_{k+1}) \in \to^D$, for all $k = 0, \ldots, N-1$.

The projection $\gamma_{\mathcal{A}}(\mathbf{T}) = q_0 \ldots q_N$ of an automaton-enabled trajectory onto the automaton states is an automaton path and the projection $\gamma_X(\mathbf{T}) = x_0 \ldots x_N$ onto the state space of system (1) is a trajectory of system (1) that follows the sequence of polyhedra defined by the automaton path.

Let $\mathbf{x} = x_0, \ldots, x_d, d \in \mathbb{Z}_+$ be a satisfying trajectory of system (1). The definition of the automaton-enabled trajectory and the construction of the dual automaton $\mathcal{A}^D$ from Section IV-A imply that there exists an automaton-enabled trajectory $\mathbf{T}$ such that $\gamma_X(\mathbf{T}) = \mathbf{x}$ and $\gamma_{\mathcal{A}}(\mathbf{T})$ is an accepting run of $\mathcal{A}^D$. Therefore, in MPC controller design, it is sufficient to search the control sequences that generate automaton-enabled trajectories. We use $\mathbf{U}_N(q, x)$ to denote the set of all control sequences of length N that produce automaton-enabled trajectories starting from (q, x) as characterized in Definition V.1. By following the standard MPC notation, we use

$$\mathbf{T}_k = (q_{0|k}, x_{0|k}) \ldots (q_{N|k}, x_{N|k}),$$

to denote a *predicted* automaton-enabled trajectory originating at (q_k, x_k), *i.e.* $q_{0|k} = q_k$, $x_{0|k} = x_k$, at time $k \in \mathbb{Z}_+$.

Problem V.2 (MPC optimization problem) At time $k \in \mathbb{Z}_+$ let $(q_k, x_k) \in \bigcup_{q \in Q^D} \{\{q\} \times \mathcal{P}_q\}$, $\{x^r_{k+i}\}_{i=0,\ldots,N-1}$, $\{u^r_{k+i}\}_{i=0,\ldots,N-1}$, $\alpha \in \mathbb{R}_+$ and $\gamma \in (0,1)$ be given. Minimize the cost function

$$\begin{aligned} C(x_k, \mathbf{u}_k) :=& \gamma^k \sum_{i=0,\ldots,N-1} L(x_{i|k}, u_{i|k}) + \\ & (1-\gamma^k)\alpha V(q_{N|k}, x_{N|k}), \end{aligned} \tag{13}$$

over all control sequences $\mathbf{u}_k = u_{0|k}, \ldots, u_{N-1|k} \in \mathbf{U}_N(q_k, x_k)$ subject to

$$x_{i+1|k} = Ax_{i|k} + Bu_{i|k}, \quad i = 0, \ldots, N-1. \tag{14}$$

The objective of the optimization is to minimize the cost with respect to the available reference state and control trajectories, while guaranteeing that the resulting trajectory reaches an accepting state. To enforce the latter part, the potential function $V(\cdot, \cdot)$ (11) is used as the terminal cost. As time progresses, the weight of the terminal cost, *i.e.* $1 - \gamma^k$, increases, which in turn guarantees that the resulting trajectory steers towards an accepting state. The value of the potential function is scaled by a constant factor $\alpha \in \mathbb{R}_+$, since the objective is to minimize the potential and trajectory cost together.

The optimization problem formulation is analogous to the classical MPC formulation, where $L(\cdot, \cdot)$, $V(\cdot, \cdot)$, and N are called the stage cost function, the terminal cost function, and the prediction horizon, respectively [13]. However, due to the definition of an automaton-enabled trajectory, there are significant differences, *e.g.* the search space ($\mathbf{U}_N(q_k, x_k)$) is not necessarily convex.

Next, we show that the optimal solution of Problem V.2 can be found by solving a set of convex optimization problems. Specifically, we propose to solve an optimization problem for each automaton path from the set

$$\begin{aligned} \mathbf{P}^N_{q_k} = \{\mathbf{q}' = q_{0|k} q_{1|k} \ldots q_{N|k} \mid q_{0|k} := q_k, \exists d \in \mathbb{Z}_+ \\ s.t.\ N \leq d, \mathbf{q} = q_{0|k} \ldots q_{d|k}, \mathbf{q} \in \mathbf{P}_{q_k}\}, \end{aligned} \tag{15}$$

where $\mathbf{P}_{q_k}$ is defined as in (9). The definition of an automaton-enabled trajectory $\mathbf{T}_k$ of horizon N (Definition V.1) implies that $\gamma_{\mathcal{A}}(\mathbf{T}_k) \in \mathbf{P}^N_{q_k}$ for any trajectory that can be produced by a control sequence $\mathbf{u} \in \mathbf{U}_N(q_k, x_k)$.

Given a finite automaton path $\mathbf{q}_k \in \mathbf{P}^N_{q_k}$, let $\mathbf{U}^{\mathbf{q}_k}_N(q_k, x_k)$ denote the set of all control sequences that produce an automaton-enabled trajectory $\mathbf{T}_k$ with $\gamma_{\mathcal{A}}(\mathbf{T}_k) = \mathbf{q}_k$. Essentially, $\mathbf{U}^{\mathbf{q}_k}_N(q_k, x_k)$ is the set of all control sequences that produce trajectories of system (1) that originate at x_k and follow the sequence of polyhedra defined by $\mathbf{q}_k$. Then, it is straightforward to see that

$$\mathbf{U}_N(q_k, x_k) = \bigcup_{\mathbf{q}_k \in \mathbf{P}^N_{q_k}} \mathbf{U}^{\mathbf{q}_k}_N(q_k, x_k). \tag{16}$$

Consider a path $\mathbf{q}_k = q_{0|k} \cdots q_{N|k} \in \mathbf{P}^N_{q_k}$ and the following optimization problem in the variables $\mathbf{u}_k = u_{0|k}, \ldots, u_{N-1|k}$:

$$\begin{aligned} & \min C(x_k, \mathbf{u}_k), \\ & \textit{subject to} \\ & x_{i|k} \in \mathcal{P}_{q_{i|k}}, \qquad i = 1, \ldots, N, \qquad (17a) \\ & u_{i|k} \in \mathbb{U}, \qquad i = 0, \ldots, N-1,, \qquad (17b) \end{aligned}$$

where $C(\cdot, \cdot)$ and $x_{i|k}, i = 1, \ldots, N$ are defined as in (13) and (14), respectively. The set of control sequences that satisfy constraints (17a) and (17b) is $\mathbf{U}^{\mathbf{q}_k}_N(q_k, x_k)$. Therefore, the optimal solution of Problem V.2 can be found by solving an optimization problem as given in (17) for each $\mathbf{q}_k \in \mathbf{P}^N_{q_k}$.

As shown above, the solution of Problem V.2 can be found by solving a set of convex optimization problems for a given prediction horizon. To guarantee that the resulting closed-loop trajectory of system (1) reaches a region $\mathcal{P}_{q_f}$, where $q_f \in F^D$; at each time-step k the prediction horizon, denoted as I_k, is determined with respect to the predicted trajectory obtained at the previous step. Specifically, the length of the observed reference trajectory, N, is used as the initial prediction horizon I_0 at time-step $k = 0$. Then, for time-step $k \geq 1$, if the predicted trajectory obtained at the previous step visits a final state at position j for the first time, $j - 1$ is used as the prediction horizon I_k. Otherwise, the same prediction horizon as in the previous time-step, I_{k-1}, is used. The following function is used to determine the prediction horizon for a given trajectory

$\mathbf{T}_k = (q_{0|k}, x_{0|k}) \ldots (q_{I_k|k}, x_{I_k|k})$:

$$I(\mathbf{T}_k) = \begin{cases} I_k & \text{if } 0 < V(q_{i|k}, x_{i|k}), \forall i = 0, \ldots, I_k \\ j-1 & \text{if } 0 < V(q_{i|k}, x_{i|k}), \forall i = 0, \ldots, j-1, \\ & \qquad\qquad V(q_{j|k}, x_{j|k}) = 0. \end{cases}$$

Adapting the prediction horizon according to function $I(\cdot)$ allows us to optimize the cost until the specification is satisfied, *i.e.* until a final automaton state is reached.

At each time step k, the proposed MPC controller solves the optimization problem (17) for each automaton path $\mathbf{q} \in \mathbf{P}_{q_k}^{I_k}$ (15), finds the optimal solution $\mathbf{u}_k^*$ among all feasible solutions of these QPs, applies the first control from $\mathbf{u}_k^*$ and computes (q_{k+1}, x_{k+1}).

Assumption V.3 The length of any satisfying trajectory of system (1) originating at x_0 is lower bounded by N.

Lemma V.4 *Suppose that Assumption IV.3 and Assumption V.3 hold, and there exists $q_0 \in Q^D$ such that $x_0 \in \mathcal{P}_{q_0}$. Then, the optimization problem given in* (17) *is feasible for some $\mathbf{q}_0 \in \mathbf{P}_{q_0}^N$ at the initial condition (q_0, x_0).*

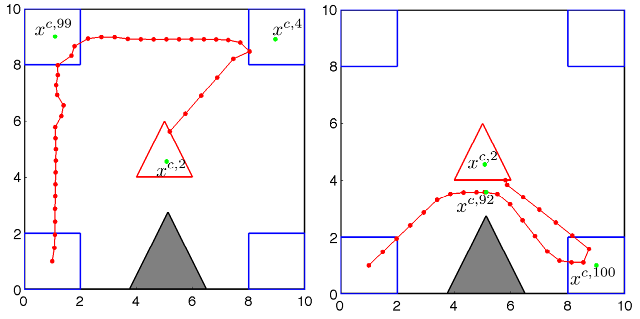
The proposed controller is recursively feasible, meaning that if Problem V.2 is feasible for the initial state at the initial time instant, then it remains feasible until the specification is satisfied, which is formally stated as:

Theorem V.5 *Suppose that Assumption IV.3 and Assumption V.3 hold, $V(\cdot,\cdot)$ and $J^T(\cdot,\cdot)$ are defined as in* (11) *and* (10)*, respectively, and there exists $q_0 \in Q^D$ such that $x_0 \in \mathcal{P}_{q_0}$. Then:*

(i) *If the optimization problem given in* (17) *is feasible for some $\mathbf{q}_k \in \mathbf{P}_{q_k}^{I_k}$ at time k for state (q_k, x_k) and $q_{k+1} \notin F^D$, then there exists $\mathbf{q}_{k+1} \in \mathbf{P}_{q_{k+1}}^{I_{k+1}}$ such that the problem is feasible for $\mathbf{q}_{k+1}$ and state (q_{k+1}, x_{k+1}).*

(ii) *The trajectory of system* (1) *produced by the closed-loop system satisfies the specification.*

VI. Case Study



(a)

(b)

Fig. 2. The trajectories of the controlled system. (a-b) The reference trajectories are generated from automaton sequences q_{99}, q_4, q_2 and q_{92}, q_{100}, q_2, respectively. The center points of the corresponding polytopes are marked with green dots.

Consider the system dynamics and specification given in Example IV.2, and a cost function defined as in (2) with

$$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, \quad R = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}. \tag{18}$$

We define reference trajectories according to a sequence of automaton states. In particular, for a given sequence of automaton states $q_0, \ldots, q_d$, we define the first $N-1$ states of the reference trajectory as $x_i^r := x^{c,0}, i = 0, \ldots, N-2$, where $x^{c,0}$ is the center of the polytope $\mathcal{P}_{q_0}$. Then, we keep an index variable j (initialized to $j = 0$), and at each time step $k \geq 0$, we generate x_{k+N-1}^r and update j according to the state x_k of the controlled system as follows:

$$[x_{k+N-1}^r, j] := \begin{cases} [x^{c,j+1}, j+1] & \text{if } x_k \in \mathcal{P}_{q_j} \\ [x_{k+N-2}^r, j] & \text{otherwise.} \end{cases} \tag{19}$$

Two system trajectories generated by the MPC controller are shown in Figure 2 (a) and (b), where the reference trajectories are generated as explained above according to the sequences of automaton states q_{99}, q_4, q_2 and q_{92}, q_{100}, q_2, respectively. For both of the experiments, the reference control sequences are defined as $u_i^r = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right], i \in \mathbb{Z}_+$, the prediction horizon (N) is 5, the scaling factor α is 1.45, and the weight constant γ is 0.95.

Note that both of the trajectories from Figure 2 satisfy the specification Φ_{ex}. The experiments show that we can use the reference trajectories to steer the closed-loop trajectory towards the desired regions, which was not possible in [8].

References

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

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

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

[4] A. Bhatia, M. Maly, L. E. Kavraki, and M. Y. Vardi, "Motion planning with complex goals," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 55 –64, sept. 2011.

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

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

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

[8] E. A. Gol and M. Lazar, "Temporal logic model predictive control for discrete-time systems," in *Hybrid Systems: Computation and Control*. ACM, 2013, pp. 343–352.

[9] X. C. Ding, M. Lazar, and C. Belta, "Receding horizon temporal logic control for finite deterministic systems," in *American Control Conference*, Montreal, Canada, 2012, pp. 715 –720.

[10] E. A. Gol, "Formal verification and controller synthesis for discrete-time systems," Ph.D. dissertation, Boston University, MA,USA, 2014.

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

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

[13] J. B. Rawlings and D. Q. Mayne, *Model predictive control theory and desing*. Nob Hill Pub, 2009.