Automata Theory Meets Approximate Dynamic Programming: Optimal Control with Temporal Logic Constraints

Ivan Papusha[†] Jie Fu[*] Ufuk Topcu[‡] Richard M. Murray[†]

***Abstract*— We investigate the synthesis of optimal controllers for continuous-time and continuous-state systems under temporal logic specifications. The specification is expressed as a deterministic, finite automaton (the specification automaton) with transition costs, and the optimal system behavior is captured by a cost function that is integrated over time. We construct a dynamic programming problem over the product of the underlying continuous-time, continuous-state system and the discrete specification automaton. To solve this dynamic program, we propose controller synthesis algorithms based on approximate dynamic programming (ADP) for both linear and nonlinear systems under temporal logic constraints. We argue that ADP allows treating the synthesis problem directly, without forming expensive discrete abstractions. We show that, for linear systems under co-safe temporal logic constraints, the ADP solution reduces to a single semidefinite program.**

I. Introduction

In this work, we address the problem of optimal control of dynamical systems under temporal logic specifications. Dynamical systems of interest to control are typically written as differential equations on a continuous state space, with inputs that can take on a continuum of values over a continuous time interval. However, temporal logic constraints that permit decidable synthesis must work with a finite or countable parameterization of time and space. As a result, a control designer must either forgo the continuous dynamics, create a discrete abstraction, or somehow re-express the temporal constraints within their optimal control framework.

Abstraction-based, hierarchical, and symbolic control methods have been proposed for continuous systems under temporal logic constraints [1–8]. These classes of methods involve three general steps: 1) abstracting the dynamical system as a discrete finite-state system, 2) synthesizing a discrete control law using a product of the specification and the abstraction, and 3) compatibly implementing the discrete control law on the original continuous system. Approximate abstractions can be developed by reachability-based computational methods, counter-example guided abstraction refinement, and sampling-based methods [1], [9–13]. However, it is well-known that the abstraction process is computationally expensive. In addition, it is difficult to ensure the optimality of a control policy designed at the abstraction level with respect to a given continuous cost function.

[†]Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA.

[*]Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA.

[‡]Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, TX

To address the issues of scalability, correctness, and optimality, Wolff et al. [14], [15] have formulated the control problem directly as a mixed-integer linear program on the system variables, which avoids a finite abstraction. This encoding method applies to a fragment of Linear Temporal Logic (LTL), while being sensitive to the accuracy of time discretization. Earlier attempts at restricting LTL semantics to work with continuous states also relied on a "flat" subset of LTL [16], resulting in the construction of a hybrid automaton, the size of which is exponential in the size of the specification formula. More recent work has concentrated on using sum-of-squares techniques and barrier certificates [17].

This paper exploits the idea that continuous-time and continuous-state systems constrained by LTL specifications can be viewed as a hybrid dynamical system through an augmentation of the continuous state space with the discrete states of the specification automaton. The resulting optimality conditions consist of mixed continuous-discrete Hamilton–Jacobi–Bellman (HJB) equations, which are difficult to solve in general. Therefore, approximate dynamic programming (ADP) can be used to approximate the value function, and to give an approximate policy. If this approximate policy satisfies the logical specifications (checked by simulation), then we can give bounds on the policy's suboptimality with respect to the objective. Otherwise, a more expressive basis must be chosen. The optimization-based ADP framework provides considerable freedom to choose a relevant basis representation, although we demonstrate that a quadratic basis is often sufficient. Under mild assumptions, an optimal policy that satisfies the specifications can be recovered if the basis is complete, see [18, Thm. 3], [19].

Our dynamic program makes use of a finite acceptance condition of the specification automaton by effectively treating controller synthesis as a kind of shortest path problem. As a result, our framework is also limited to a subset of LTL for its temporal specification language—in this case the fragment is called *co-safe* LTL [20]. Importantly, co-safe LTL admits its own automaton construction method, e.g., [21], which is more efficient than Büchi automata constructions for general LTL [22]. An earlier similar proposed architecture used the product of a co-safe LTL specification automaton and a discrete abstraction of nonlinear robot dynamics with a sampling-based planner [23]. By comparison, we treat the hybrid dynamics directly, and restrict to piecewise linear systems obviating the need for nonlinear random tree path planning. Specifically, the main parameters under the designer's control are the value function bases used, rather than the fidelity of the discretization of time or space.

We state the specific problem and describe the co-safe LTL fragment in §II. We define the product between the continuous-time, continuous-space dynamics and the discrete automaton corresponding to the co-safe LTL specification, and then write down the dynamic program in §III. We describe an optimization-based framework for ADP in this context in §IV, and give examples of how to implement the optimization problem as a semidefinite program for linear systems in §V. We conclude in §VI.

II. Problem Description

We consider a continuous-time and continuous-state dynamical system on $\mathbf{R}^n$. This system is given by

$$\dot{x} = f(x,u), \quad x(0) = x_0, \tag{1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbf{R}^n$ and $u(t) \in \mathcal{U} \subseteq \mathbf{R}^m$ are the state and control signals at time t. For simplicity, we restrict f to be a Lipschitz continuous function of (x,u), and the control input u to be a piecewise right-continuous function of time, with finitely many discontinuities on any finite time interval. These conditions are not always required, but they ensure the existence and uniqueness of solutions, and are meant to prevent Zeno behavior.

The system (1) is constrained to satisfy a specification on the discrete behavior obtained from its continuous trajectory. First, let AP be a finite set of atomic propositions, which are logical predicates that hold true when $x(t)$ is in a particular region. Then, define a labeling function $L : \mathcal{X} \to \Sigma = 2^{AP}$, which maps a continuous state $x \in \mathcal{X}$ to the finite subset of atomic propositions that evaluate to true at x. This function partitions the continuous space $\mathcal{X}$ into regions that share the same truth values in AP. The labeling function also links the continuous system with its *discrete behavior*. In the following definition, $\phi(x_0, [0,T], u)$ refers to the trajectory of the continuous system with initial condition x_0 under the control input $u(t)$ over the time interval $[0,T]$.

Definition. *Let $t_0, t_1, \ldots, t_N$ be times, such that*

- $0 = t_0 < t_1 < \cdots < t_N = T$,
- $L(x(t)) = L(x(t_k))$, $t_k \le t < t_{k+1}$, $k = 0, \ldots, N$,
- $L(x(t_k^-)) \neq L(x(t_k^+))$, $k = 0, \ldots, N$.

The discrete behavior, *denoted $\mathcal{B}(\phi(x_0,[0,T],u))$, is the discrete word $\sigma_0\sigma_1\ldots\sigma_{N-1} \in \Sigma^*$, where $\sigma_k = L(x(t_k))$.*

A specification on the discrete behavior can be written as a co-safe LTL formula over the finite set of atomic propositions (for a comprehensive description of the syntax and semantics of LTL, the reader is referred to [24], [25]). A co-safe LTL formula is an LTL formula where every satisfying word has a finite good prefix[1] [20]. We restrict to such formulas to take advantage of the expressiveness of temporal logic for specifying optimal control problems without imposing infinite Büchi acceptance conditions [22]. We give examples of appropriate specifications in §V.

[1]Given a word $w \in \Sigma^*$ and $v \in \Sigma^*$, v is a prefix of w if and only if $w = vu$ for some $u \in \Sigma^*$. The word u is called the suffix of w.

Given a co-safe LTL specification φ over the set of atomic propositions AP, there exists a corresponding deterministic finite-state automaton (DFA) $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F\rangle$, where Q is a finite set of states (modes), $\Sigma = 2^{AP}$ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is a *deterministic* transition function such that when the symbol $\sigma \in \Sigma$ is read at state q, the automaton makes a deterministic transition to state $\delta(q,\sigma) = q'$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final, or *accepting* states. The transition function is extended to a sequence of symbols, or a *word* $w = \sigma_0\sigma_1\ldots \in \Sigma^*$, in the usual way: $\delta(q, \sigma_0 v) = \delta(\delta(q,\sigma_0), v)$ for $\sigma_0 \in \Sigma$ and $v \in \Sigma^*$. We say that the finite word w satisfies φ if and only if $\delta(q_0, w) \in F$. The set of words satisfying φ is the *language* of the automaton $\mathcal{A}_\varphi$, denoted $\mathcal{L}(\mathcal{A}_\varphi)$.

The discrete behavior encodes the sequence of labels visited by the state as it moves along its continuous trajectory. Specifically, the atomic propositions are evaluated only at the times when the label changes value. Thus a trajectory $\phi(x_0, [0,T], u)$ satisfies an LTL specification φ if and only if its discrete behavior is in the language $\mathcal{L}(\mathcal{A}_\varphi)$. The optimal control problem is formulated as follows.

Problem 1. *Consider the system* (1)*, a co-safe LTL specification φ, and a final state $x_f \in \mathcal{X}$. Design a control law u that minimizes the cost function*

$$J = \int_0^T \ell(x(\tau), u(\tau))\, d\tau + \sum_{k=0}^{N} s(x(t_k), q(t_k^-), q(t_k^+)) \tag{2}$$

subject to the constraints that $\mathcal{B}(\phi(x_0,[0,T],u)) \in \mathcal{L}(\mathcal{A}_\varphi)$ and $x(T) = x_f$.

Here, $\ell : \mathcal{X} \times \mathcal{U} \to \mathbf{R}$ is a continuous loss function, and $s : \mathcal{X} \times Q \times Q \to \mathbf{R}$ is the cost to transition between two states of the automaton whenever such a transition is allowed. The final state $x(T) = x_f$ is also specified. Similar problems have been studied in existing work [3], [8], [13], [15], [26–28]. The novelty of our approach is twofold. First, we show that this co-safe LTL problem can be cast as an optimal hybrid control problem, and second, we use approximate dynamic programming to synthesize a suboptimal controller with guaranteed performance bounds.

III. Product Formulation

To solve Problem 1, we first augment the continuous state space $\mathcal{X}$ with the discrete state space Q of the specification automaton $\mathcal{A}_\varphi$ to obtain a hybrid system. The construction of $\mathcal{A}_\varphi$ for a specific formula φ can be automated with existing tools [21], [22]. We show that the optimal control problem constrained by a co-safe LTL specification φ is a dynamic programming problem over the product space $\mathcal{X} \times Q$.

In this setting, the hybrid state at time t is an ordered pair $(x(t), q(t)) \in \mathcal{X} \times Q$. Evolution of the continuous-state component $x(t)$ is governed by the original system flow (1), while evolution of the discrete component $q(t)$ is governed by the appropriate deterministic transition of the specification automaton. Such a transition is initiated when the continuous state crosses a boundary between two labeled

regions. Specifically, we consider the following product hybrid system:

Definition. *The* product system $H = \langle Q, \mathcal{X}, E, f, R, G\rangle$ *is an internally forced hybrid system, where*

- Q *is the set of discrete states (modes) of* $\mathcal{A}_\varphi$,
- $\mathcal{X} \subseteq \mathbf{R}^n$ *is the set of continuous states,*
- $E \subseteq Q \times \Sigma \times Q$ *is a set of discrete transitions, where* $e = (q, \sigma, q') \in E$ *if and only if* $\delta(q, \sigma) = q'$,
- $f : \mathcal{X} \times \mathcal{U} \to \mathbf{R}^n$ *is the continuous vector field given by* (1),
- $R = \{R_q \mid q \in Q\}$ *is a collection of regions, where*

$$R_{q,\sigma} = \{x \in \mathcal{X} \mid \exists q' \in Q : (q', \sigma, q) \in E, \text{ and } L(x) = \sigma\}, \quad q \in Q, \sigma \in \Sigma,$$
$$R_q = \bigcup_{\sigma \in \Sigma} R_{q,\sigma}, \quad q \in Q,$$

- $G = \{G_e \mid e \in E\}$ *is a collection of guards, where*

$$G_e = \{x \in \mathbf{bd}\, R_{q,\sigma} \mid \delta(q, L(x)) = q'\},$$

for all $e = (q, \sigma, q') \in E$.

Each region R_q refers to the continuous states $x \in \mathcal{X}$ that are reachable while the automaton is in or transitions to mode q. For each discrete mode q, the continuous state evolves inside R_q until it enters a guard region $G_{(q,\sigma,q')}$ and a discrete transition to mode q' is made.

We can solve the optimal control problem with dynamic programming by ensuring that the optimal value function is zero at every accepting state of the automaton. Let $V^\star : \mathcal{X} \times Q \to \mathbf{R}$ be the optimal cost-to-go in (2), with $V^\star(x_0, q_0)$ denoting the optimal objective value when starting at the initial condition (x_0, q_0), subject to the discrete behavior specification and final condition $x(T) = x_f$. For simplicity, we assume that $V^\star$ has no explicit dependence on t, which corresponds to searching for a stationary policy, although this assumption can be relaxed at the expense of having to choose a time-varying basis when searching for an approximate value function later. In this setting, the cost-to-go satisfies a collection of mixed continuous-discrete Hamilton–Jacobi–Bellman (HJB) equations,

$$0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u) \right\}, \quad \forall x \in R_q,\ \forall q \in Q, \tag{3}$$

$$V^\star(x,q) = \min_{q'} \{V^\star(x,q') + s(x,q,q')\}, \quad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E, \tag{4}$$

$$0 = V^\star(x_f, q_f), \quad \forall q_f \in F. \tag{5}$$

Equation (3) says that $V^\star(x,q)$ is an optimal cost-to-go inside the regions where the label remains constant. The next equation (4) is a shortest-path equality that must hold at every continuous state x where a discrete state transition to a different label can happen. Finally, the boundary equation (5) fixes the value function.

We can interpret these HJB conditions intuitively as a single-sink shortest-path problem on a directed weighted graph, where nodes with the same label are treated together and the weights are the incremental costs $\ell(x,u)dt$ or the discrete transition costs $s(x,q,q')$ (Fig. 1). As long as the continuous state evolves within the same labeled region, the value function is subject to the optimality condition associated with the region that contains that state. As a result, the continuous-state condition (3) must hold on the interior nodes (white), while the discrete-state switching condition (4) must hold at the boundary nodes (black).

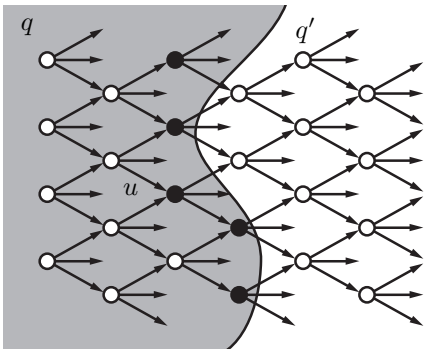

Fig. 1. Finite state interpretation of HJB conditions (3)–(5)

The graph interpretation also clarifies why automata derived from co-safe LTL specifications fit within the dynamic programming framework but not automata derived from more general temporal logics: the semantics of general LTL are over infinite execution traces, and require Büchi automata whose acceptance conditions do not readily translate to a single-sink shortest-path problem. Nevertheless, we view the co-safe restriction as a strength, rather than weakness, because co-safe LTL is highly expressive for practical control problems, and because the solution methods we describe in the next section are efficient for many classes of problems, relatively simple to implement, and can be readily automated.

IV. Lower bounds on the optimal cost

Let $V^\star$ be a value function satisfying the hybrid HJB conditions (3)–(5), and suppose V is another function that satisfies the following inequalities,

$$0 \le \frac{\partial V(x,q)}{\partial x} \cdot f(x,u) + \ell(x,u), \quad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q, \tag{6}$$

$$0 \le V(x,q') - V(x,q) + s(x,q,q'), \quad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E, \tag{7}$$

$$0 = V(x_f, q_f), \quad \forall q_f \in F. \tag{8}$$

Then $V(x_0,q_0) \le V^\star(x_0,q_0)$. This approach is motivated by [18], [26], [29]. The inequalities (6)–(8) characterize a set of optimal value function under-estimators, among which is the optimal value function $V^\star$ itself. The difference between the equalities (3)–(5) and the inequalities (6)–(8) is the removal of the minimum operators in favor of semi-infinite constraints and the addition of pointwise inequalities.

The strength of using inequalities to search over value function under-estimators, instead of solving the HJB equations directly, lies in an ability to come up with *approximate*

value functions and ADP policies whose suboptimality can be quantified, e.g., [30]. The ADP method is enabled by the fact that we can come up with sufficient conditions that imply (6)–(8), and relies on finding the largest approximate value function that is a pointwise under-estimate of $V^\star$. Thus we solve the problem

$$\begin{array}{ll} \text{maximize} & V(x_0, q_0) \\ \text{subject to} & (6), (7), \text{ and } (8) \end{array} \tag{9}$$

over the variables parameterizing the under-estimate V.

The approximate value function V is written as a sum of basis functions as follows,

$$V(x, q) = \sum_{i=1}^{n_q} w_{i,q} \phi_{i,q}(x),$$

where $\phi_{i,q} : \mathcal{X} \to \mathbf{R}$ are given basis functions, and the coefficients $w_{i,q}$, $i = 1, \ldots, n_q$ are the variables. Given an approximate value function V, an approximately optimal control law can be implemented as

$$u(x, q) \in \operatorname*{argmin}_{u \in \mathcal{U}} \left\{ \frac{\partial V(x, q)}{\partial x} \cdot f(x, u) + \ell(x, u) \right\}.$$

Switching between different discrete modes is autonomous, and the scaling with problem size can be controlled with an appropriate parameterization of V. For more information on ADP, see the references [31], [32]. The goal of the rest of this section is to describe how to solve the optimization problem (9) in specific instances.

A. Linear quadratic systems

In this section we describe how to search for an approximate value function for the linear system

$$\dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0,$$

with a quadratic continuous cost

$$\ell(x, u) = x^T Q x + u^T R u, \quad Q \succeq 0, \quad R \succ 0,$$

and a constant switching cost

$$s(x, q, q') = \begin{cases} \xi & \text{if } q \neq q', \\ 0 & \text{otherwise,} \end{cases}$$

where $\xi > 0$ is a given positive constant.

We parameterize the approximate value function as a quadratic. For each $q \in Q$, let

$$V(x, q) = x^T P_q x + 2 r_q^T x + t_q, \quad \text{for all } x \in \mathcal{X}, \tag{10}$$

where $P_q = P_q^T \in \mathbf{R}^{n \times n}$, $r_q \in \mathbf{R}^n$ and $t_q \in \mathbf{R}$ are the variables of the parameterization. This parameterization is linear in (P_q, r_q, t_q) and corresponds to choosing basis functions $\phi_{i,q}(x)$ that are quadratic, linear, and constant in the components of x, respectively.

With the approximate value function in (10), the objective function is then

$$V(x_0, q_0) = x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0}.$$

For the constraints, condition (6) is the same as

$$0 \leq \begin{bmatrix} x \\ u \\ 1 \end{bmatrix}^T \begin{bmatrix} A^T P_q + P_q A + Q & P_q B & A^T r_q \\ B^T P_q & R & B^T r_q \\ r_q^T A & r_q^T B & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ 1 \end{bmatrix} \quad \forall x \in R_q, \ \forall u \in \mathcal{U}, \ \forall q \in Q, \tag{11}$$

i.e., it is a collection of $|Q|$ semi-infinite constraints indexed by the continuous variables $x \in \mathcal{X}$ and $u \in \mathcal{U}$, one for each $q \in Q$. If R_q is a quadratically representable set, e.g., ellipsoids or halfspaces, then we directly use the S-procedure to obtain a finite number of sufficient conditions for (11) to hold [30]. Otherwise if R_q is not quadratically representable, then two approaches can be used. One approach is exact: Given state $q \in Q$, we partition R_q to a finite set of quadratically representable sets $R_q = \bigcup_{i=1}^{N_q} R_q^i$ and enforce constraint (11) in state q to each R_q^i with parameterization (P_q^i, r_q^i, t_q^i). Note that such a partition is possible if the set of states satisfying each atomic proposition is quadratically representable. Alternatively, we can overapproximate R_q with a quadratically representable set. In both cases, $\mathcal{U}$ can be overapproximated by a quadratically representable set.

Similarly, the inequality (7) is enforced over each guard region by the collection of $|E|$ semi-infinite constraints

$$0 \leq \begin{bmatrix} x \\ 1 \end{bmatrix}^T \begin{bmatrix} P_{q'} - P_q & r_{q'} - r_q \\ r_{q'}^T - r_q^T & t_{q'} - t_q + \xi \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} \quad \forall x \in G_e, \ \forall e \in E. \tag{12}$$

Once again, the S-procedure translates the semi-infinite constraints into a finite sufficient condition if the guard regions G_e are quadratically representable. The guard regions are quadratically representable when each atomic proposition corresponds to a quadratically representable set in $\mathcal{X}$.

Finally, the condition (8) relates P_{q_f}, r_{q_f}, and t_{q_f} via the $|F|$ linear equality constraints

$$0 = x_f^T P_{q_f} x_f + 2 r_{q_f}^T x_f + t_{q_f}, \quad \forall q_f \in F, \tag{13}$$

where x_f is the given fixed final state.

B. Nonlinear systems

For general nonlinear flows, it is appropriate to choose a more expressive value function approximation. A typical approach would use a radial basis function (RBF) basis [33], [34, §12],

$$V(x, q) = \sum_{i=1}^{m} w_{i,q} \exp\left(-\frac{\|x - c_i\|_2^2}{a_i^2} \right), \tag{14}$$

where $\{c_i\}_{i=1}^m$ is a finite set of center points in $\mathbf{R}^n$ chosen to sample the continuous state space $\mathcal{X}$, and $\{a_i\}_{i=1}^m$ are positive constants that define the RBF widths. The same set of basis functions can be used everywhere, or alternatively the RBF centers can be chosen to have most of their support over the regions R_q appropriate to each state $q \in Q$.

With the approximate value function in (14), the objective and constraints (6)–(8) lead to the optimization problem

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{m} w_{i,q}\phi_{i,q}(x_0), \\
\text{subject to} \quad & 0 \leq \sum_{i=1}^{m} w_{i,q}\left(\frac{\partial \phi_{i,q}(x)}{\partial x}\cdot f(x,u)\right) + \ell(x,u), \\
& \qquad \forall x \in R_q,\ \forall u \in \mathcal{U},\ \forall q \in Q, \\
& 0 \leq \sum_{i=1}^{m} (w_{i,q'} - w_{i,q})\phi_i(x) + s(x,q,q'), \\
& \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E, \\
& 0 = \sum_{i=1}^{m} w_{i,q_f}\phi_i(x_f), \quad \forall q_f \in F,
\end{aligned}$$

with variables $w_{i,q}$. The switching cost function s can be defined as in §IV-A.

This semi-infinite LP has a finite number $m \times |Q|$ of variables but an infinite number of constraints. A general solution approach is to sample the constraints, and then solve the finite linear program. The derivation of probably approximately correct (PAC) bounds on the needed number of samples requires careful study of the specific dynamics and costs in the optimization problem.

V. Examples

A. Linear quadratic systems with halfspace labels

We consider the linear quadratic system on $\mathcal{X} = \mathbf{R}^2$ from §IV-A with the specific parameters

$$\begin{gathered}
A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \\
Q = I, \quad R = 1, \quad \xi = 1, \\
x_0 = (0.5, 0), \quad x_f = (0,0).
\end{gathered}$$

Let $AP = \{a, b\}$ consist of atomic propositions that are true whenever the continuous state enters a specific region,

$$\begin{aligned}
a = \text{True} &\iff (x(t) \in R_A), \quad R_A = \{x \in \mathbf{R}^2 \mid x_1 \leq 1\} \\
b = \text{True} &\iff (x(t) \in R_B), \quad R_B = \{x \in \mathbf{R}^2 \mid x_1 > 1\}.
\end{aligned}$$

Note that R_A and R_B are closed halfspaces and the interface between them is the line $G = \{x \in \mathbf{R}^2 \mid x_1 = 1\}$. We define the labeling function $L : \mathbf{R}^2 \to \{A, B\} \subseteq \Sigma = 2^{AP}$, where $A = \{a\}$ and $B = \{b\}$. The goal is to satisfy the specification

$$\varphi_1 = \Diamond A \wedge \Diamond B,$$

where $\Diamond$ is the LTL "eventually" operator.

The automaton that accepts this specification consists of four states and is shown in Fig. 2. The discrete behavior accepted by this automaton is any trajectory that eventually visits each of the regions R_A and R_B.

The guard regions are defined as

$$\begin{gathered}
G_{(q_0,A,q_1)} = R_A, \quad G_{(q_0,B,q_2)} = R_B, \\
G_{(q_1,B,q_2)} = G_{(q_2,A,q_2)} = G.
\end{gathered}$$

Note that the direction of crossing the guard region is encoded by the allowed DFA transitions.

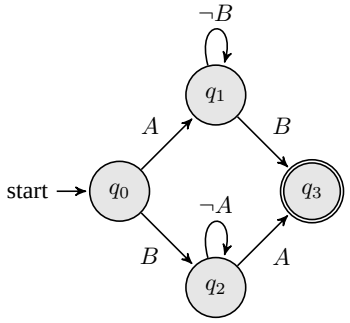

Fig. 2. Automaton $\mathcal{A}_{\varphi_1}$ for $\varphi_1 = \Diamond A \wedge \Diamond B$

Guided by the automaton, we mechanically apply the S-procedure to obtain the semidefinite program

$$\begin{aligned}
\text{maximize} \quad & x_0^T P_{q_0} x_0 + 2 r_{q_0}^T x_0 + t_{q_0} \\
\text{subject to} \quad & (11), (12), \text{ and } (13),
\end{aligned}$$

with variables $P_q = P_q^T \in \mathbf{R}^{2\times 2}$, $r_q \in \mathbf{R}^2$, $t_q \in \mathbf{R}$, $q \in Q = \{q_0, \ldots, q_3\}$, and eight additional variables coming from the S-procedure. With $x_f = 0$, the final constraint (13) translates to $t_{q_3} = 0$.

The semidefinite program was solved using SDPT3 and resulted, within numerical accuracy, in the following value function:

$$\begin{aligned}
P_q^\star &= \begin{bmatrix} 22.314 & -28.142 \\ -28.142 & 38.799 \end{bmatrix}, \quad q = q_0, \ldots, q_3, \\
r_q^\star &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad q = q_0, \ldots, q_3, \\
t_q^\star &= \begin{cases} 2, & q = q_0 \\ 1, & q = q_1, q_2 \\ 0, & q = q_3. \end{cases}
\end{aligned}$$

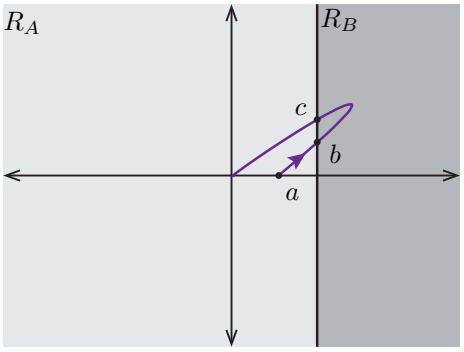

Fig. 3. Approximately minimum cost path satisfying φ_1 with initial condition $x_0 = (0.5, 0)$. To satisfy φ_1, the trajectory must leave R_A and visit R_B.

Note that the shape of $V(\cdot, q)$ is the same for every state q of the automaton, with the only difference being the offset t_q. The policy implied by this value function is illustrated in Fig. 3 for the specific hybrid execution trace with initial condition $x_0 = (0.5, 0)$. The reader is invited to follow the figure as we interpret the execution:

1) The path $x(t)$ starts at the point a with initial condition $x_0 = (0.5, 0)$ and automaton state q_0.
2) The automaton makes an immediate transition to q_1, because $L(x_0) = A$ and the value function is lower for this discrete state. The continuous dynamics follow the negative gradient of $V(\cdot, q_1)$.
3) At point b, the automaton transitions to q_3. The continuous dynamics go down the gradient of $V(\cdot, q_3)$ in the segment of the path between b and c.
4) At point c, the automaton is already in its accepting state q_3. The continuous dynamics continue to follow the negative gradient of $V(\cdot, q_3)$ to reach $x_f = 0$.

B. More complex specification

We now consider three regions, R_A, R_B, and R_C with the slightly more complex specification

$$\varphi_2 = (A \to \Diamond B) \wedge (C \to \Diamond B) \wedge (\Diamond A \vee \Diamond C).$$

This specification ensures that either R_A or R_C must be reached, after which the system must eventually visit R_B. The automaton for this specification is shown in Fig. 4. Depending on the accrued continuous and transition costs, there is a choice to go left or right in Fig. 5.

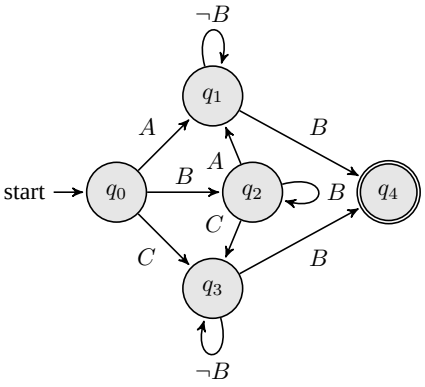

Fig. 4. Automaton $\mathcal{A}_{\varphi_2}$ for $\varphi_2 = (A \to \Diamond B) \wedge (C \to \Diamond B) \wedge (\Diamond A \vee \Diamond C)$

We form the semidefinite program as before to obtain five approximate value functions $V(\cdot, q)$, one for each $q \in Q = \{q_0, \ldots, q_4\}$ in the automaton. This time, we plot the execution for two initial conditions $x_0 = (-0.5, -0.5)$, whose path (abc) goes right, and $x_0 = (-0.5, 0)$, whose path (def) goes left. See Fig. 5.

To interpret this policy, it is valuable to compare the spectra of the closed loop matrix

$$A_q^{\text{cl}} = A - BR^{-1}B^T P_q^\star$$

in the initial mode $q = q_0$ against the accepting mode $q = q_4$,

$$\lambda(A_{q_0}^{\text{cl}}) = \{0.786 \pm 1.144j\}, \quad \lambda(A_{q_4}^{\text{cl}}) = \{-1 \pm j\}.$$

In the initial state q_0, the closed loop eigenvalues are unstable, while they are stable in the final state q_4. Our procedure therefore recovers the requirement of φ_2 that a trajectory starting near the origin in region R_B must go away to visit another region, and eventually transition to an accepting state of the automaton before being allowed back to $x_f = 0$.

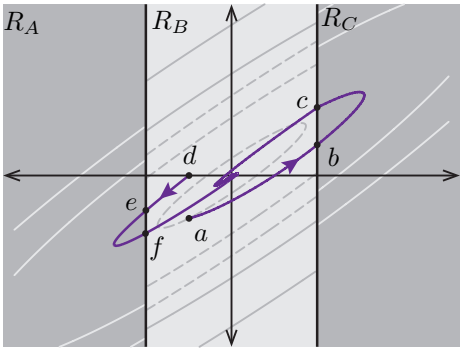

Fig. 5. Approximately minimum cost paths satisfying φ_2, and level sets of the value function active in each region: the path abc with initial condition $x_0 = (-0.5, -0.5)$ satisfies φ_2 by visiting R_C, while the path def with initial condition $x_0 = (-0.5, 0)$ satisfies φ_2 by visiting R_A. Note that the level sets of $V(\cdot, q_2)$ (solid, inside R_B) have a subtle tilt and magnitude shift compared to $V(\cdot, q_4)$ (dashed, inside R_B), which allows for the excursion away from the origin required by φ_2.

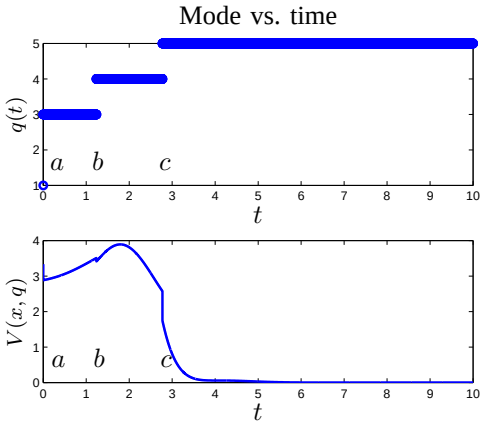

Fig. 6. State of $\mathcal{A}_{\varphi_2}$ and value function along the path abc going right.

VI. Conclusion

In this work, we approached the problem of optimal control under co-safe LTL constraints with approximate dynamic programming (ADP). The approximate policy is given by following a sequence of value functions over a hybrid state space, where the continuous component comes from the continuous-time and continuous-state dynamics of the system, and the discrete component comes from the specification automaton. For linear dynamical systems with quadratic cost functions, we used the specification automaton to construct a semidefinite program that gives the suboptimal policy. This procedure does not rely on discretizing the time/state space or formulating non-convex optimization problems. The proposed framework can be incorporated as a building block in other approximate control methods for scalable synthesis of systems with LTL specifications.

Our ADP approach is limited to a co-safe subset of LTL specifications that admit deterministic and finite automaton representations. Extensions to the general class of Büchi automaton representations with continuous-time dynamics are subjects of future work. Other directions include sampling the semi-infinite constraints, and exploring the limitations of different value function bases for general nonlinear systems.

ACKNOWLEDGMENTS

This work was supported in part by a Department of Defense NDSEG Fellowship, the Boeing company, AFRL FA8650-15-C-2546, ONR N000141310778, ARO W911NF-15-1-0592, NSF 1550212, DARPA W911NF-16-1-0001, and ONR N00014-15-IP-00052. The authors wish to acknowledge R. Ehlers for helpful discussions, and the anonymous reviewers for helping improve this paper.

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] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[3] L. C. G. J. M. Habets and C. Belta, "Temporal logic control for piecewise-affine hybrid systems on polytopes," in *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Jul. 2010, pp. 195–202.

[4] J. Liu and N. Ozay, "Abstraction, discretization, and robustness in temporal logic control of dynamical systems," in *International Conference on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2014, pp. 293–302.

[5] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, Jul. 2000.

[6] P. Tabuada, "Symbolic control of linear systems based on symbolic subsystems," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 1003–1013, Jun. 2006.

[7] S. L. Smith, J. Tůmova, C. Belta, and D. Rus, "Optimal path planning for surveilance with temporal-logic constraints," *International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, Dec. 2011.

[8] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, Nov. 2012.

[9] E. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald, "Abstraction and counterexample-guided refinement in model checking of hybrid systems," *International Journal of Foundations of Computer Science*, vol. 14, no. 04, pp. 583–604, 2003.

[10] G. Reißig, "Computing abstractions of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 11, pp. 2583–2598, 2011.

[11] J. Fu and H. G. Tanner, "Bottom-up symbolic control: Attractor-based planning and behavior synthesis," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3142–3155, 2013.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications," in *American Control Conference (ACC)*, Jun. 2012, pp. 735–742.

[13] N. Kariotoglou, S. Summers, T. Summers, M. Kamgarpour, and J. Lygeros, "Approximate dynamic programming for stochastic reachability," in *European Control Conference (ECC)*, Jul. 2013, pp. 584–589.

[14] E. M. Wolff and R. M. Murray, "Optimal control of nonlinear systems with temporal logic specifications," in *International Symposium on Robotics Research (ISRR)*, 2013.

[15] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2013, pp. 4332–4339.

[16] G. E. Fainekos, S. G. Loizou, and G. J. Pappas, "Translating temporal logic to controller specifications," in *IEEE Conference on Decision and Control (CDC)*, Dec. 2006, pp. 899–904.

[17] T. Wongpiromsarn, U. Topcu, and A. Lamperski, "Automata theory meets barrier certificates: Temporal logic verification of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, Dec. 2015.

[18] A. Rantzer, "Dynamic programming via convex optimization," in *IFAC World Congress*, 1999, pp. 491–496.

[19] R. B. Vinter and R. M. Lewis, "A necessary and sufficient condition for optimality of dynamic programming type, making no a priori assumptions on the controls," *SIAM Journal on Control and Optimization*, vol. 16, no. 4, pp. 571–583, 1978.

[20] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, Nov. 2001.

[21] T. Latvala, "Efficient model checking of safety properties," in *International SPIN Workshop on Model Checking of Software*, ser. Lecture Notes in Computer Science, T. Ball and S. K. Rajamani, Eds. Springer, 2003, vol. 2648, pp. 74–88.

[22] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *International Conference on Computer Aided Verification (CAV'01)*, ser. Lecture Notes in Computer Science, G. Berry, H. Comon, and A. Finkel, Eds., vol. 2102. Paris, France: Springer, Jul. 2001, pp. 53–65. [Online]. Available: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PS/Cav01go.ps

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

[24] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[25] C. Baier and J.-P. Katoen, *Principles of Model Checking*, ser. Representation and Mind. MIT Press, 2008.

[26] S. Hedlund and A. Rantzer, "Optimal control of hybrid systems," in *IEEE Conference on Decision and Control (CDC)*, vol. 4, 1999, pp. 3972–3977.

[27] ——, "Convex dynamic programming for hybrid systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1536–1540, 2002.

[28] X. Xu and P. J. Antsaklis, "Optimal control of switched systems based on parameterization of the switching instants," *IEEE Transactions on Automatic Control*, vol. 49, no. 1, pp. 2–16, 2004.

[29] D. P. de Farias and B. V. Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[30] Y. Wang and S. P. Boyd, "Performance bounds for linear stochastic control," *Systems & Control Letters*, vol. 58, no. 3, pp. 178–182, 2009.

[31] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[32] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, ser. Wiley Series in Probability and Statistics. Wiley-Interscience, 2007.

[33] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[34] E. Lavretsky and K. A. Wise, *Robust and Adaptive Control: with Aerospace Applications*, ser. Advanced Textbooks in Control and Signal Processing. Springer, 2013.

[35] T. H. Summers, K. Kunz, N. Kariotoglou, M. Kamgarpour, S. Summers, and J. Lygeros, "Approximate dynamic programming via sum of squares programming," in *European Control Conference (ECC)*, Jul. 2013, pp. 191–197.

[36] M. Johansson and A. Rantzer, "Computation of piecewise quadratic Lyapunov functions for hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 555–559, Apr. 1998.