MDP Optimal Control under Temporal Logic Constraints

Xu Chu Ding Stephen L. Smith Calin Belta Daniela Rus

Abstract— **In this paper, we develop a method to automatically generate a control policy for a dynamical system modeled as a Markov Decision Process (MDP). The control specification is given as a Linear Temporal Logic (LTL) formula over a set of propositions defined on the states of the MDP. We synthesize a control policy such that the MDP satisfies the given specification almost surely, if such a policy exists. In addition, we designate an "optimizing proposition" to be repeatedly satisfied, and we formulate a novel optimization criterion in terms of minimizing the expected cost in between satisfactions of this proposition. We propose a sufficient condition for a policy to be optimal, and develop a dynamic programming algorithm that synthesizes a policy that is optimal under some conditions, and sub-optimal otherwise. This problem is motivated by robotic applications requiring persistent tasks, such as environmental monitoring or data gathering, to be performed.**

I. Introduction

Recently, many works [1]–[4] have proposed using temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [5], as specification languages for control systems. Such logics are appealing because they have well defined syntax and semantics, which can be easily used to specify complex behavior. In particular, in LTL, it is possible to specify persistent tasks, *e.g.,* "Visit regions A, then B, and then C, infinitely often. Never enter B unless coming directly from D." In addition, off-the-shelf model checking algorithms [5] and temporal logic game strategies [6] can be used to verify the correctness of system trajectories and to synthesize provably correct control strategies.

The existing works focusing on LTL assume that a finite model of the system is available and the current state can be precisely determined. If the control model is deterministic (*i.e.,* at each state, an available control enables exactly one transition), control strategies from specifications given as LTL formulas can be found through simple adaptations of off-the-shelf model checking algorithms [7]. If the control is non-deterministic (an available control at a state enables one of several transitions, and their probabilities are not known), the control problem from an LTL specification can be mapped to the solution of a Büchi or GR(1) game if the specification is restricted to a fragment of LTL [1], [8]. If the probabilities of the enabled transitions at each state are known, the control problem reduces to finding a control policy for a Markov Decision Process (MDP) such that a probabilistic temporal logic formula is satisfied [9].

X. C. Ding and C. Belta are with Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA (email: {xcding; cbelta}@bu.edu). S. L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (email: stephen.smith@uwaterloo.ca). D. Rus is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (email: rus@csail.mit.edu). This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, and NSF CNS-0834260.

By adapting methods from probabilistic model-checking [9]–[11], we have recently developed frameworks for deriving MDP control policies from LTL formulas [12], which is related to a number of other approaches [13], [14] that address the problem of synthesizing control policies for MDPs subject to LTL satisfaction constraints. In all of the above approaches, a control policy is designed to maximize the probability of satisfying a given LTL formula. However, no attempt has been made so far to optimize the long-term behavior of the system, while enforcing LTL satisfaction guarantees. Such an objective is often critical in many applications, such as surveillance, persistent monitoring, and pickup delivery tasks, where an autonomous agent must repeatedly visit some areas in an environment and the time in between revisits should be minimized.

As far as we know, this paper is the first attempt to compute an optimal control policy for a dynamical system modeled as an MDP, while satisfying temporal logic constraints. This work begins to bridge the gap between our prior work on MDP control policies maximizing the probability of satisfying an LTL formula [12] and optimal path planning under LTL constraints [15]. We consider LTL formulas defined over a set of propositions assigned to the states of the MDP. We synthesize a control policy such that the MDP satisfies the specification almost surely. In addition, over all such policies, we minimize the expected cost between satisfying instances of an "optimizing proposition."

The main contribution of this paper is two-fold. First, we formulate the above MDP optimization problem in terms of minimizing the average cost per *cycle*, where a cycle is defined between successive satisfaction of the optimizing proposition. We present a novel connection between this problem and the well-known average cost per stage problem. Second, we incorporate the LTL constraints and obtain a sufficient condition for a policy to be optimal. We present a dynamic programming algorithm that under some restrictions synthesizes an optimal control policy, and a sub-optimal policy otherwise.

Due to space constraints, we omit all proofs in the paper and the complexity analysis of the approach. They can be found in the technical report [16].

II. Preliminaries

A. Linear Temporal Logic

We employ Linear Temporal Logic (LTL) to describe MDP control specifications. A detailed description of the syntax and semantics of LTL is beyond the scope of this paper and can be found in [5], [10]. Roughly, an LTL formula is built up from a set of atomic propositions Π, standard Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and temporal operators $\bigcirc$ (next), $\mathcal{U}$ (until), $\diamond$

(eventually), $\square$ (always). More expressivity can be achieved by combining the above temporal and Boolean operators. An LTL formula ϕ over Π is evaluated over infinite words $o = o_0 o_1 \dots$ in 2^{Π} (*i.e., o* either satisfies or violates ϕ).

Definition II.1 (Deterministic Rabin Automaton)**.** *A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$, where (i) Q is a finite set of states; (ii) Σ is a set of inputs (alphabet); (iii) $\delta : Q \times \Sigma \to Q$ is the transition function; (iv) $q_0 \in Q$ is the initial state; and (v) $F = \{(L(1), K(1)), \dots, (L(M), K(M))\}$ is a set of pairs of sets of states such that $L(i), K(i) \subseteq Q$ for all $i = 1, \dots, M$.*

A run of a Rabin automaton $\mathcal{R}$, denoted by $r_{\mathcal{R}} = q_0 q_1 \dots$, is an infinite sequence of states in $\mathcal{R}$ such that for each $i \geq 0$, $q_{i+1} \in \delta(q_i, \alpha)$ for some $\alpha \in \Sigma$. A run $r_{\mathcal{R}}$ is *accepting* if there exists a pair $(L(i), K(i))$ such that set $L(i)$ is visited by $r_{\mathcal{R}}$ only finitely often and set $K(i)$ is visited by $r_{\mathcal{R}}$ infinitely often. For any LTL formula ϕ over Π, one can construct a DRA with input alphabet $\Sigma = 2^{\Pi}$ accepting all and only words over Π that satisfy ϕ (see *e.g.,* [10]). We refer readers to [10] and references therein for algorithms and to freely available implementations, such as [17], to translate an LTL formula to a corresponding DRA.

B. Markov Decision Process and probability measure

Definition II.2 (Labeled Markov Decision Process)**.** *A labeled Markov decision process (MDP) is a tuple $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$, where $S = \{1, \dots, n\}$ is a finite set of states; U is a finite set of controls (actions) (with slight abuse of notations we also define the function $U(i)$, where $i \in S$ and $U(i) \subseteq U$ to represent the available controls at state i); $P : S \times U \times S \to [0, 1]$ is the transition probability function such that for all $i \in S$, $\sum_{j \in S} P(i, u, j) = 1$ if $u \in U(i)$, and $P(i, u, j) = 0$ if $u \notin U(i)$; $s_0 \in S$ is the initial state; Π is a set of atomic propositions; $\mathcal{L} : S \to 2^{\Pi}$ is a labeling function and $g : S \times U \to \mathbb{R}^+$ is such that $g(i, u)$ is the expected (positive) cost when control $u \in U(i)$ is taken at state i.*

We define a control function $\mu : S \to U$ such that $\mu(i) \in U(i)$ for all $i \in S$. An infinite sequence of control functions $M = \{\mu_0, \mu_1, \dots\}$ is called a *policy*. One can use a policy to resolve all non-deterministic choices in an MDP by applying the action $\mu_k(s_k)$ at state s_k. Given an initial state s_0, an infinite sequence $r^M_{\mathcal{M}} = s_0 s_1 \dots$ on $\mathcal{M}$ generated under a policy M is called a *path* on $\mathcal{M}$ if $P(s_k, \mu_k(s_k), s_{k+1}) > 0$ for all k. The index k of a path is called *stage*. If $\mu_k = \mu$ for all k, then we call it a *stationary* policy and we denote it simply as μ. A stationary policy μ induces a Markov chain where its set of states is S and the transition probability from state i to j is $P(i, \mu(i), j)$.

We define $\mathrm{Paths}^M_{\mathcal{M}}$ as the set of all paths of $\mathcal{M}$ under a policy M. We can then define a probability measure $\mathrm{Pr}^M_{\mathcal{M}}$ over the set $\mathrm{Paths}^M_{\mathcal{M}}$. The definition of this measure can be found in a text in probabilistic model checking, such as [10, Ch. 10]. With this probability measure, we can define the probability that an MDP $\mathcal{M}$ under a policy M satisfies an LTL formula ϕ. A path $r^M_{\mathcal{M}} = s_0 s_1 \dots$ deterministically generates a word $o = o_0 o_1 \dots$, where $o_i = \mathcal{L}(s_i)$ for all i. With a slight abuse of notation, we denote $\mathcal{L}(r^M_{\mathcal{M}})$ as the word generated by $r^M_{\mathcal{M}}$. We denote $o \vDash \phi$ if word o satisfies ϕ. Given an LTL formula ϕ, one can show that the set $\{r^M_{\mathcal{M}} \in \mathrm{Paths}^M_{\mathcal{M}} : \mathcal{L}(r^M_{\mathcal{M}}) \vDash \phi\}$ is measurable. We define

$$\mathrm{Pr}^M_{\mathcal{M}}(\phi) := \mathrm{Pr}^M_{\mathcal{M}}\{r^M_{\mathcal{M}} \in \mathrm{Paths}^M_{\mathcal{M}} : \mathcal{L}(r^M_{\mathcal{M}}) \vDash \phi\} \qquad (1)$$

as the probability of satisfying ϕ for $\mathcal{M}$ under M. See [10] for more details about probability measures on MDPs under a policy and measurability of LTL formulas.

III. Problem Formulation

Consider a weighted MDP $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$ and an LTL formula ϕ over Π. As proposed in [15], we assume that formula ϕ is of the form:

$$\phi = \square \diamond \pi \wedge \psi, \qquad (2)$$

where the atomic proposition $\pi \in \Pi$ is called the *optimizing proposition* and ψ is an arbitrary LTL formula. In other words, ϕ requires that ψ be satisfied and π be satisfied infinitely often. We assume that there exists at least one policy M of $\mathcal{M}$ such that $\mathcal{M}$ under M satisfies ϕ almost surely, *i.e.,* $\mathrm{Pr}^M_{\mathcal{M}}(\phi) = 1$ (in this case we simply say M satisfies ϕ almost surely).

We let $\mathbb{M}$ be the set of all policies and $\mathbb{M}_\phi$ be the set of all policies satisfying ϕ almost surely. Note that if there exists a control policy satisfying ϕ almost surely, then there typically exist many (possibly infinite number of) such policies.

We would like to obtain the optimal policy such that ϕ is almost surely satisfied, and the expected cost in between visiting a state satisfying π is minimized. To formalize this, we first denote $S_\pi = \{i \in S, \pi \in \mathcal{L}(i)\}$ (*i.e.,* the states where atomic proposition π is true). We say that each visit to set S_π *completes a cycle*. Thus, starting at the initial state, the finite path reaching S_π for the first time is the first cycle; the finite path that starts after the completion of the first cycle and ends with revisiting S_π for the second time is the second cycle, and so on. Given a path $r^M_{\mathcal{M}} = s_0 s_1 \dots$, we use $C(r^M_{\mathcal{M}}, N)$ to denote the cycle index up to stage N, which is defined as the total number of cycles completed in N stages plus 1 (*i.e.,* the cycle index starts with 1 at the initial state).

The main problem that we consider in this paper is to find a policy that almost surely satisfies ϕ and minimizes the average cost per cycle (ACPC) starting from the initial state s_0. Formally, we have:

Problem III.1. *Find a policy $M = \{\mu_0, \mu_1, \dots\}$ over $\mathbb{M}_\phi$ that minimizes*

$$J(s_0) = \limsup_{N \to \infty} E\left\{ \frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{C(r^M_{\mathcal{M}}, N)} \right\}, \qquad (3)$$

where $E\{\cdot\}$ denotes the expected value.

Prob. III.1 is related to the standard average cost per stage (ACPS) problem, which consists of minimizing

$$J^s(s_0) = \limsup_{N \to \infty} E\left\{ \frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{N} \right\}, \qquad (4)$$

over $\mathbb{M}$, with the noted difference that the right-hand-side (RHS) of (4) is divided by the index of stages instead of cycles. The ACPS problem has been widely studied in the dynamic programming community, without the constraint of satisfying temporal logic formulas.

The ACPC cost function we consider in this paper is relevant for probabilistic abstractions and practical applications, where the cost of controls can represent the time, energy, or fuel required to apply controls at each state. In particular, it is a suitable performance measure for persistent tasks, which can be specified by LTL formulas. For example, in a data gathering mission [15], an agent is required to repeatedly gather and upload data. We can assign π to the data upload locations and a solution to Prob. III.1 minimizes the expected cost in between data upload. In such cases, the ACPS cost function does not translate to a meaningful performance criterion. In fact, a policy minimizing (4) may even produce an infinite cost in (3). Nevertheless, we will make the connection between the ACPS and the ACPC problems in Sec. IV.

IV. Solving the average cost per cycle problem

A. Optimality conditions for ACPS problems

In this section, we recall some known results on the ACPS problem, namely finding a policy over $\mathbb{M}$ that minimizes J^s in (4). The reader interested in more details is referred to standard texts (*e.g.,* [18], [19]) and references therein.

Definition IV.1 (Weak Accessibility Condition)**.** *An MDP $\mathcal{M}$ is said to satisfy the Weak Accessibility (WA) condition if there exist $S_r \subseteq S$, such that (i) there exists a stationary policy where j is reachable from i for any $i, j \in S_r$, and (ii) states in $S \setminus S_r$ are transient under all stationary policies.*

MDP $\mathcal{M}$ is called *single-chain* (or *weakly-communicating*) if it satisfies the WA condition. If $\mathcal{M}$ satisfies the WA condition with $S_r = S$, then $\mathcal{M}$ is called *communicating*. For a stationary policy, it induces a Markov chain with a set of recurrent classes. A state that does not belong to any recurrent class is called *transient*. A stationary policy μ is called *unichain* if the Markov chain induced by μ contains one recurrent class (and a possible set of transient states). If every stationary policy is unichain, $\mathcal{M}$ is called unichain.

Recall that the set of states of $\mathcal{M}$ is denoted by $\{1, \ldots, n\}$. For each stationary policy μ, we use P_μ to denote the transition probability matrix: $P_\mu(i, j) = P(i, \mu(i), j)$. Define vector g_μ where $g_\mu(i) = g(i, \mu(i))$. For each stationary policy μ, we can obtain a *gain-bias* pair (J^s_μ, h^s_μ), where

$$J^s_\mu = P^*_\mu g_\mu, \qquad h^s_\mu = H^s_\mu g_\mu \tag{5}$$

with

$$P^*_\mu = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} (P_\mu)^k, \qquad H_\mu = (I - P_\mu + P^*_\mu)^{-1} - P^*_\mu. \tag{6}$$

The vector $J^s_\mu = [J^s_\mu(1), \ldots, J^s_\mu(n)]^{\mathrm{T}}$ is such that $J^s_\mu(i)$ is the ACPS starting at initial state i under policy μ. Note that the limit in (6) exists for any stochastic matrix P_μ, and P^*_μ is stochastic. Therefore, the $\limsup$ in (4) can be replaced by the limit for a stationary policy. Moreover, (J^s_μ, h^s_μ) satisfies

$$J^s_\mu = P_\mu J^s_\mu, \qquad J^s_\mu + h^s_\mu = g_\mu + P_\mu h^s_\mu. \tag{7}$$

By noting that

$$h^s_\mu + v^s_\mu = P_\mu v^s_\mu, \tag{8}$$

for some vector v^s_μ, we see that $(J^s_\mu, h^s_\mu, v^s_\mu)$ is the solution of $3n$ linear equations with $3n$ unknowns.

It has been shown that, there exists a stationary optimal policy $\mu^\star$ minimizing (4) over all policies. Furthermore, If $\mathcal{M}$ is single-chain or communicating, the optimal ACPS does not depend on the initial state, *i.e.,* $J^s_{\mu^\star}(i) = \lambda$ for all $i \in S$ and the gain-bias pair is $(\lambda\mathbf{1}, h^s)$, where $\mathbf{1}$ is a vector of all 1s. In this case, the following optimality condition can be derived from the Bellman's equations of optimality for ACPS problems (see, *e.g.,* [18]):

$$\lambda\mathbf{1} + h^s \leq g_\mu + P_\mu h^s \tag{9}$$

for all $\mu \in \mathbb{M}$, where $\leq$ is component-wise.

B. Optimality conditions for ACPC problems

Now we derive optimality conditions for ACPC problems, without considering the satisfaction constraint, *i.e.,* we do not limit the set of polices to $\mathbb{M}_\phi$ at the moment. We consider the following problem:

Problem IV.2. *Given a communicating MDP $\mathcal{M}$ and a set S_π, find a policy μ over $\mathbb{M}$ that minimizes* (3)*.*

Note that, for reasons that will become clear in Sec. V, we assume in Prob. IV.2 that the MDP is communicating. However, it is possible to generalize the results in this section to an MDP that is not communicating.

We limit our attention to stationary policies. We will show that, similar to the majority of problems in dynamic programming, there exist optimal stationary policies, thus it is sufficient to consider only stationary policies. For such policies, we use the following notion of *proper policies*, which is the same as the one used in stochastic shortest path problems (see *e.g.,* [18]).

Definition IV.3 (Proper Polices)**.** *We say a stationary policy μ is proper if, under μ, there is positive probability that the set S_π will be reached after a finite number of stages, regardless of the initial state.*

We denote $J_\mu = [J_\mu(1), \ldots, J_\mu(n)]^{\mathrm{T}}$ where $J_\mu(i)$ is the ACPC in (3) starting from state i under policy μ. If policy μ is improper, then there exist some states $i \in S$ that can never reach S_π. In this case, since $g(i, u)$ is positive for all i, u, we can immediately see that $J_\mu(i) = \infty$. We will first consider only proper policies.

Without loss of generality, we assume that $S_\pi = \{1, \ldots, m\}$ (*i.e.,* states $m+1, \ldots, n$ are not in S_π). Given a proper policy μ, we obtain its transition matrix P_μ as described in Sec. IV-A. Our goal is to express J_μ in terms of P_μ, similar to (5) in the ACPS case. To achieve this, we first compute the probability that $j \in S_\pi$ is the first state visited in S_π after leaving from a state $i \in S$ by applying policy μ. We denote this probability by $\widetilde{P}(i, \mu, j)$. We can obtain this probability for all $i \in S$ and $j \in S_\pi$ using the following:

Proposition IV.4. *$\widetilde{P}(i, \mu, j)$ satisfies*

$$\widetilde{P}(i, \mu, j) = \sum_{k=m+1}^{n} P(i, \mu(i), k)\widetilde{P}(k, \mu(k), j) + P(i, \mu(i), j). \tag{10}$$

We now define a $n \times n$ matrix $\widetilde{P}_\mu$ such that

$$\widetilde{P}_\mu(i,j) = \begin{cases} \widetilde{P}(i,\mu,j) & \text{if } j \in S_\pi \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

We can immediately see that $\widetilde{P}_\mu$ is a stochastic matrix, *i.e.,* $\sum_{j=1}^n \widetilde{P}(i,\mu,j) = 1$. More precisely, $\sum_{j=1}^m \widetilde{P}(i,\mu,j) = 1$ since $\widetilde{P}(i,\mu,j) = 0$ for all $j = m+1,\ldots,n$.

Using (10), we can express $\widetilde{P}_\mu$ in a matrix equation in terms of P_μ. To do this, we need to first define two $n \times n$ matrices from P_μ as follows:

$$\overleftarrow{P}_\mu(i,j) = \begin{cases} P_\mu(i,j) & \text{if } j \in S_\pi \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

$$\overrightarrow{P}_\mu(i,j) = \begin{cases} P_\mu(i,j) & \text{if } j \notin S_\pi \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

We can see that matrix P_μ is "split" into $\overleftarrow{P}_\mu$ and $\overrightarrow{P}_\mu$, *i.e.,* $P_\mu = \overleftarrow{P}_\mu + \overrightarrow{P}_\mu$.

Proposition IV.5. *If a policy μ is proper, then matrix $I - \overrightarrow{P}_\mu$ is non-singular.*

We can then write (10) as the following matrix equation:

$$\widetilde{P}_\mu = \overrightarrow{P}_\mu \widetilde{P}_\mu + \overleftarrow{P}_\mu. \quad (14)$$

Since $I - \overrightarrow{P}_\mu$ is invertible, we have

$$\widetilde{P}_\mu = (I - \overrightarrow{P}_\mu)^{-1} \overleftarrow{P}_\mu. \quad (15)$$

Note that (14) and (15) do not depend on the ordering of the states of $\mathcal{M}$, *i.e.,* S_π does not need to be equal to $\{1,\ldots,m\}$.

Next, we give an expression for the expected cost of reaching S_π from $i \in S$ under μ (if $i \in S_\pi$, this is the expected cost of reaching S_π again), and denote it as $\tilde{g}(i,\mu)$.

Proposition IV.6. *$\tilde{g}(i,\mu)$ satisfies*

$$\tilde{g}(i,\mu) = \sum_{k=m+1}^{n} P(i,\mu(i),k)\tilde{g}(k,\mu) + g(i,\mu(i)). \quad (16)$$

We define $\tilde{g}_\mu$ such that $\tilde{g}_\mu(i) = \tilde{g}(i,\mu)$, and note that (16) can be written as follows:

$$\begin{aligned} \tilde{g}_\mu &= \overrightarrow{P}_\mu \tilde{g}_\mu + g_\mu \\ \tilde{g}_\mu &= (I - \overrightarrow{P}_\mu)^{-1} g_\mu, \end{aligned} \quad (17)$$

where g_μ is defined in Sec. IV-A.

We can now express the ACPC J_μ in terms of $\widetilde{P}_\mu$ and $\tilde{g}_\mu$. Observe that, starting from i, the expected cost of the first cycle is $\tilde{g}_\mu(i)$; the expected cost of the second cycle is $\sum_{j=1}^m \widetilde{P}_\mu(i,\mu,j)\tilde{g}_\mu(j)$; the expected cost of the third cycle is $\sum_{j=1}^m \sum_{k=1}^m \widetilde{P}_\mu(i,\mu,j)\widetilde{P}_\mu(j,\mu,k)\tilde{g}_\mu(k)$; and so on. Therefore

$$J_\mu = \limsup_{C\to\infty} \frac{1}{C} \sum_{k=0}^{C-1} (\widetilde{P}_\mu)^k \tilde{g}_\mu, \quad (18)$$

where C represents the cycle count. Since $\widetilde{P}_\mu$ is a stochastic matrix, the $\limsup$ in (18) can be replaced by the limit, and we have

$$J_\mu = \lim_{C\to\infty} \frac{1}{C} \sum_{k=0}^{C-1} (\widetilde{P}_\mu)^k \tilde{g}_\mu = \widetilde{P}_\mu^* \tilde{g}_\mu, \quad (19)$$

where P^* for a stochastic matrix P is defined in (6).

We can now make a connection between Prob. IV.2 and the ACPS problem. Each proper policy μ of $\mathcal{M}$ can be mapped to a policy $\tilde{\mu}$ with transition matrix $P_{\tilde{\mu}} := \widetilde{P}_\mu$ and vector of costs $g_{\tilde{\mu}} := \tilde{g}_\mu$, and we have

$$J_\mu = J^s_{\tilde{\mu}}. \quad (20)$$

Moreover, we define $h_\mu := h^s_{\tilde{\mu}}$. Together with J_μ, pair (J_μ, h_μ) can be seen as the gain-bias pair for the ACPC problem. We denote the set of all polices that can be mapped to a proper policy as $\mathbb{M}_{\tilde{\mu}}$. We see that a proper policy minimizing the ACPC maps to a policy over $\mathbb{M}_{\tilde{\mu}}$ minimizing the ACPS.

The by-product of the above analysis is that, if μ is proper, then $J_\mu(i)$ is finite for all i, since $\widetilde{P}^*_\mu$ is stochastic and $g_\mu(i)$ is finite for all i. We now show that, among stationary policies, it is sufficient to consider only proper policies.

Proposition IV.7. *Assume μ to be an improper policy. If $\mathcal{M}$ is communicating, then there exists a proper policy μ' such that $J_{\mu'}(i) \leq J_\mu(i)$ for all $i = 1,\ldots,n$, with strict inequality for at least one i.*

Using the connection to the ACPS problem, we have:

Proposition IV.8. *If $\mathcal{M}$ is communicating, the optimal ACPC policy over stationary policies is independent of the initial state.*

The above result means that the gain-bias pair for the optimal policy is $(\lambda\mathbf{1}, h)$. This is not surprising, as it mirrors the result for a communicating MDP in the ACPS problem. Essentially, transient behavior does not matter in the long run so the optimal cost is the same for any initial state.

Using inequality in the case when the optimal cost is the same for all initial states (9), policy $\tilde{\mu}^\star$ with the ACPS gain-bias pair $(\lambda\mathbf{1}, h^s_{\tilde{\mu}^\star})$ satisfying for all $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$:

$$\lambda\mathbf{1} + h^s_{\tilde{\mu}^\star} \leq g_{\tilde{\mu}} + P_{\tilde{\mu}} h^s_{\tilde{\mu}^\star} \quad (21)$$

is optimal. Equivalently, $\mu^\star$ that maps to $\tilde{\mu}^\star$ is optimal over all proper policies. The following proposition shows that $\mu^\star$ is optimal over all policies in $\mathbb{M}$, stationary or not.

Proposition IV.9. *The proper policy $\mu^\star$ that maps to $\tilde{\mu}^\star$ satisfying* (21) *is optimal over* $\mathbb{M}$.

Unfortunately, it is not clear how to find the optimal policy from (21) except by searching through all policies in $\mathbb{M}_{\tilde{\mu}}$. This exhaustive search is not feasible for reasonably large problems. Instead, we would like equations in the form Bellman's equations (see *e.g.,* [18]), where optimizations can be carried out independently at each state.

To circumvent this difficulty, we need to express the gain-bias pair (J_μ, h_μ), which is equal to $(J^s_{\tilde{\mu}}, h^s_{\tilde{\mu}})$, in terms of μ. From (7), we have

$$J_\mu = P_{\tilde{\mu}} J_\mu, \qquad J_\mu + h_\mu = g_{\tilde{\mu}} + P_{\tilde{\mu}} h_\mu.$$

By manipulating the above equations, we can now show that J_μ and h_μ can be expressed in terms of μ (analogous to (7)) instead of $\tilde{\mu}$ via the following proposition:

Proposition IV.10. *We have:*

$$J_\mu = P_\mu J_\mu, \qquad J_\mu + h_\mu = g_\mu + P_\mu h_\mu + \overrightarrow{P}_\mu J_\mu. \quad (22)$$

Moreover, we have:

$$(I - \overrightarrow{P}_\mu)h_\mu + v_\mu = P_\mu v_\mu, \tag{23}$$

for some vector v_μ.

From Prop. IV.10, similar to the ACPS problem, (J_μ, h_μ, v_μ) can be solved together by a linear system of $3n$ equations and $3n$ unknowns. The insight gained when simplifying J_μ and h_μ in terms of μ motivates us to propose the following optimality condition for an optimal policy.

Proposition IV.11. *The policy $\mu^\star$ with gain-bias pair $(\lambda\mathbf{1}, h)$ satisfying*

$$\lambda + h(i) = \min_{u \in U(i)} \left[g(i,u) + \sum_{j=1}^{n} P(i,u,j)h(j) + \lambda \sum_{j=m+1}^{n} P(i,u,j) \right], \tag{24}$$

for all $i = 1, \ldots, n$, is the optimal policy minimizing (3) *over all policies in $\mathbb{M}$.*

We will present an algorithm that uses Prop. IV.11 to find the optimal policy in the next section. Note that, unlike (21), (24) can be checked for any policy μ by finding the minimum for all states $i = 1, \ldots, n$, which is significantly easier than searching over all proper policies.

V. Synthesizing the optimal policy under LTL constraints

In this section we outline an approach for Prob. III.1. We aim for a computational framework, which in combination with results of [12] produces a policy that both maximizes the satisfaction probability and optimizes the long-term performance of the system, using results from Sec. IV.

A. Automata-theoretical approach to LTL control synthesis

Our approach proceeds by converting the LTL formula ϕ to a DRA as defined in Def. II.1. We denote the resulting DRA as $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ with $F = \{(L(1), K(1)), \ldots, (L(M), K(M))\}$ where $L(i), K(i) \subseteq Q$ for all $i = 1, \ldots, M$. We can now obtain an MDP as the product of a labeled MDP $\mathcal{M}$ and a DRA $\mathcal{R}_\phi$, which captures all paths of $\mathcal{M}$ satisfying ϕ (see, *e.g.*, [10], [11]).

Definition V.1 (Product MDP)**.** *The product MDP $\mathcal{M} \times \mathcal{R}_\phi$ between a labeled MDP $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$ and a DRA $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ is obtained from a tuple $\mathcal{P} = (S_\mathcal{P}, U, P_\mathcal{P}, s_{\mathcal{P}0}, F_\mathcal{P}, S_{\mathcal{P}\pi}, g_\mathcal{P})$, where*

(i) $S_\mathcal{P} = S \times Q$ is a set of states;

(ii) U is a set of controls inherited from $\mathcal{M}$ and we define $U_\mathcal{P}((s,q)) = U(s)$;

(iii) $P_\mathcal{P}$ gives the transition probabilities:

$$P_\mathcal{P}((s,q), u, (s', q')) = \begin{cases} P(s,u,s') & \text{if } q' = \delta(q, \mathcal{L}(s)) \\ 0 & \text{otherwise;} \end{cases}$$

(iv) $s_{\mathcal{P}0} = (s_0, q_0)$ is the initial state;

(v) $F_\mathcal{P} = \{(L_\mathcal{P}(1), K_\mathcal{P}(1)), \ldots, (L_\mathcal{P}(M), K_\mathcal{P}(M))\}$ where $L_\mathcal{P}(i) = S \times L(i)$, $K_\mathcal{P}(i) = S \times K(i)$, for $i = 1, \ldots, M$;

(vi) $S_{\mathcal{P}\pi}$ is the set of states in $S_\mathcal{P}$ for which proposition π is satisfied. Thus, $S_{\mathcal{P}\pi} = S_\pi \times Q$;

(vii) $g_\mathcal{P}((s,q),u) = g(s,u)$ for all $(s,q) \in S_\mathcal{P}$;

Note that some states of $S_\mathcal{P}$ may be unreachable and therefore have no control available. After removing those states (via a simple graph search), P is a valid MDP and is the desired product MDP. With a slight abuse of notation we still denote the product MDP as $\mathcal{P}$ and always assume that unreachable states are removed.

There is an one-to-one correspondence between a path $s_0 s_1, \ldots$ on $\mathcal{M}$ and a path $(s_0, q_0)(s_1, q_1)\ldots$ on $\mathcal{P}$. Moreover, from the definition of $g_\mathcal{P}$, the costs along these two paths are the same. The product MDP is constructed so that, given a path $(s_0, q_0)(s_1, q_1)\ldots$, the corresponding path $s_0 s_1 \ldots$ on $\mathcal{M}$ generates a word satisfying ϕ if and only if, there exists $(L_\mathcal{P}, K_\mathcal{P}) \in F_\mathcal{P}$ such that the set $K_\mathcal{P}$ is visited infinitely often and $L_\mathcal{P}$ finitely often (see *e.g.*, [10]).

A similar one-to-one correspondence exists for policies:

Definition V.2 (Inducing a policy from $\mathcal{P}$)**.** *Given policy $M_\mathcal{P} = \{\mu_0^\mathcal{P}, \mu_1^\mathcal{P}, \ldots\}$ on $\mathcal{P}$, where $\mu_k^\mathcal{P}((s,q)) \in U_\mathcal{P}((s,q))$, it induces policy $M = \{\mu_0, \mu_1, \ldots\}$ on $\mathcal{M}$ by setting $\mu_k(s_k) = \mu_k^\mathcal{P}((s_k, q_k))$ for all k. We denote $M_\mathcal{P}|_\mathcal{M}$ as the policy induced by $M_\mathcal{P}$, and use the same notation for a set of policies.*

An induced policy can be implemented on $\mathcal{M}$ by simply keeping track of its current state on $\mathcal{P}$. Note that a stationary policy on $\mathcal{P}$ induces a non-stationary policy on $\mathcal{M}$. From the one-to-one correspondence between paths and the equivalence of their costs, the expected cost in (3) from initial state s_0 under $M_\mathcal{P}|_\mathcal{M}$ is equal to the expected cost from initial state (s_0, q_0) under $M_\mathcal{P}$.

For each pair of states $(L_\mathcal{P}, K_\mathcal{P}) \in F_\mathcal{P}$, we can obtain a set of accepting maximal end components (AMEC):

Definition V.3 (Accepting Maximal End Components)**.** *Given $(L_\mathcal{P}, K_\mathcal{P}) \in F_\mathcal{P}$, an end component $\mathcal{C}$ is a communicating MDP $(S_\mathcal{C}, U_\mathcal{C}, P_\mathcal{C}, K_\mathcal{C}, S_{\mathcal{C}\pi}, g_\mathcal{C})$ such that $S_\mathcal{C} \subseteq S_\mathcal{P}$, $U_\mathcal{C} \subseteq U_\mathcal{P}$, $U_\mathcal{C}(i) \subseteq U(i)$ for all $i \in S_\mathcal{C}$, $K_\mathcal{C} = S_\mathcal{C} \cap K_\mathcal{P}$, $S_{\mathcal{C}\pi} = S_\mathcal{C} \cap S_{\mathcal{P}\pi}$, and $g_\mathcal{C}(i,u) = g_\mathcal{P}(i,u)$ if $i \in S_\mathcal{C}$, $u \in U_\mathcal{C}(i)$. If $P(i,u,j) > 0$ for any $i \in S_\mathcal{C}$ and $u \in U_\mathcal{C}(i)$, then $j \in S_\mathcal{C}$, in which case $P_\mathcal{C}(i,u,j) = P(i,u,j)$. An accepting maximal end components (AMEC) is the largest such end component such that $K_\mathcal{C} \neq \emptyset$ and $S_\mathcal{C} \cap L_\mathcal{P} = \emptyset$.*

An algorithm to obtain all AMECs for a product MDP was provided in [10]. Note that, an AMEC always contains at least one state in $K_\mathcal{P}$ and no state in $L_\mathcal{P}$. Moreover, it is "absorbing" in the sense that the state does not leave an AMEC once entered. From probabilistic model checking, a policy $M = M_\mathcal{P}|_\mathcal{M}$ almost surely satisfies ϕ (*i.e.*, $M \in \mathbb{M}_\phi$) if and only if, there exists an AMEC $\mathcal{C}$ such that the probability of reaching $S_\mathcal{C}$ from initial state (s_0, q_0) under policy $M_\mathcal{P}$ is 1. If such a policy $M_\mathcal{P}$ exists for an AMEC $\mathcal{C}$, we call $\mathcal{C}$ *reachable*. In [12], such a policy was found by dynamic programming or solving a linear program. For states inside $\mathcal{C}$, since $\mathcal{C}$ itself is a communicating MDP, a policy (not unique) can be easily constructed such that a state in $K_\mathcal{C}$ is infinitely often visited, satisfying the LTL specification.

B. Optimizing the long-term performance of the MDP

For a control policy designed to satisfy an LTL formula, the system behavior outside an AMEC is transient, while the behavior inside an AMEC is long-term. The policies obtained in [12]–[14] essentially disregard the behavior inside an AMEC, because, from the verification point of view, the behavior inside an AMEC is for the most part irrelevant, as long as a state in $K_{\mathcal{P}}$ is visited infinitely often. We now aim to optimize the long-term behavior of the MDP with respect to the ACPC cost function, while enforcing the satisfaction constraint. Since each AMEC is a communicating MDP, we can use results in Sec. IV-B to help obtaining a solution. Our approach consists of the following steps:

(i) Convert formula ϕ to a DRA $\mathcal{R}_\phi$ and obtain the product MDP $\mathcal{P}$ between $\mathcal{M}$ and $\mathcal{R}_\phi$;

(ii) Obtain the set of reachable AMECs, denoted as $\mathcal{A}$;

(iii) For each $\mathcal{C} \in \mathcal{A}$: Find a stationary policy $\mu^\star_{\to\mathcal{C}}(i)$, defined for $i \in S \setminus S_\mathcal{C}$, that reaches $S_\mathcal{C}$ with probability 1 ($\mu^\star_{\to\mathcal{C}}$ is guaranteed to exist and obtained as in [12]); Find a stationary policy $\mu^\star_{\circlearrowright\mathcal{C}}(i)$, defined for $i \in S_\mathcal{C}$, which minimizes (3) for MDP $\mathcal{C}$ and set $S_{\mathcal{C}\pi}$ with optimal cost $\lambda_\mathcal{C}$, while satisfying the LTL constraint (how to find this policy will be explained later); Define

$$\mu^\star_\mathcal{C} = \begin{cases} \mu^\star_{\to\mathcal{C}}(i) & \text{if } i \notin S_\mathcal{C} \\ \mu^\star_{\circlearrowright\mathcal{C}}(i) & \text{if } i \in S_\mathcal{C} \end{cases}; \tag{25}$$

(iv) We find the solution to Prob. III.1 by

$$J^\star(s_0) = \min_{\mathcal{C}\in\mathcal{A}} \lambda_\mathcal{C}, \tag{26}$$

and the optimal policy is $\mu^\star_{\mathcal{C}^\star}|_\mathcal{M}$, where $\mathcal{C}^\star$ is the AMEC attaining the minimum in (26).

We now provide the sufficient conditions for a policy $\mu^\star_{\circlearrowright\mathcal{C}}$ to be optimal. Moreover, if an optimal policy $\mu^\star_{\circlearrowright\mathcal{C}}$ can be obtained for each $\mathcal{C}$, we show that the above procedure indeed gives the optimal solution to Prob. III.1.

Proposition V.4. *For each $\mathcal{C} \in \mathcal{A}$, let $\mu^\star_\mathcal{C}$ to be constructed as in* (25)*, where $\mu^\star_{\circlearrowright\mathcal{C}}$ is a stationary policy satisfying two optimality conditions: (i) its ACPC gain-bias pair is $(\lambda_\mathcal{C}\mathbf{1}, h)$, where*

$$\lambda_\mathcal{C} + h(i) = \min_{u\in U_\mathcal{C}(i)} \Bigg[g_\mathcal{C}(i,u) + \sum_{j\in S_\mathcal{C}} P(i,u,j)h(j) + \lambda_\mathcal{C} \sum_{j\notin S_{\mathcal{C}\pi}} P(i,u,j) \Bigg], \tag{27}$$

for all $i \in S_\mathcal{C}$, and (ii) there exists a state of $K_\mathcal{C}$ in each recurrent class of $\mu^\star_{\circlearrowright\mathcal{C}}$. Then the optimal cost for Prob. III.1 is $J^\star(s_0) = \min_{\mathcal{C}\in\mathcal{A}} \lambda_\mathcal{C}$, and the optimal policy is $\mu^\star_{\mathcal{C}^\star}|_\mathcal{M}$, where $\mathcal{C}^\star$ is the AMEC attaining this minimum.

We can relax the optimality conditions for $\mu^\star_{\circlearrowright\mathcal{C}}$ in Prop. V.4 and require that there exist a state $i \in K_\mathcal{C}$ in one recurrent class of $\mu^\star_{\circlearrowright\mathcal{C}}$. For such a policy, we can construct a policy such that it has one recurrent class containing state i, with the same ACPC cost at each state. This construction is identical to a similar procedure for ACPS problems when the MDP is communicating (see [18, p. 203]). We can then use (25) to obtain the optimal policy $\mu^\star_\mathcal{C}$ for $\mathcal{C}$.

We now present an algorithm (see Alg. 1) that iteratively updates the policy in an attempt to find one that satisfies the optimality conditions given in Prop. V.4, for a given $\mathcal{C} \in \mathcal{A}$. Note that Alg. 1 is similar in nature to policy iteration algorithms for ACPS problems.

Algorithm 1 : Policy iteration algorithm for ACPC

```
Input: $\mathcal{C} = (S_\mathcal{C}, U_\mathcal{C}, P_\mathcal{C}, K_\mathcal{C}, S_{\mathcal{C}\pi}, g_\mathcal{C})$
Output: Policy $\mu_{\circlearrowright\mathcal{C}}$
1: Initialize $\mu^0$ to a proper policy containing $K_\mathcal{C}$ in its recurrent
   classes (such a policy can always be constructed since $\mathcal{C}$ is
   communicating)
2: repeat
3:     Given $\mu^k$, compute $J_{\mu^k}$ and $h_{\mu^k}$ with (22) and (23)
4:     Compute for all $i \in S_\mathcal{C}$:
           $\bar{U}(i) = \arg\min_{u\in U_\mathcal{C}(i)} \sum_{j\in S_\mathcal{C}} P(i,u,j)J_{\mu^k}(j)$    (28)
5:     if $\mu^k(i) \in \bar{U}(i)$ for all $i \in S_\mathcal{C}$ then
6:         Compute, for all $i \in S_\mathcal{C}$:
               $\bar{M}(i) = \arg\min_{u\in\bar{U}(i)} \Big[ g_\mathcal{C}(i,u) + \sum_{j\in S_\mathcal{C}} P(i,u,j)h_{\mu^k}(j) + \sum_{j\notin S_{\mathcal{C}\pi}} P(i,u,j)J_{\mu^k}(j) \Big]$    (29)
7:         Find $\mu^{k+1}$ such that $\mu^{k+1}(i) \in \bar{M}(i)$ for all $i \in S_\mathcal{C}$, and
           contains a state of $K_\mathcal{C}$ in its recurrent classes. If one does
           not exist. Return: $\mu^k$ with “not optimal”
8:     else
9:         Find $\mu^{k+1}$ such that $\mu^{k+1}(i) \in \bar{U}(i)$ for all $i \in S_\mathcal{C}$, and
           contains a state of $K_\mathcal{C}$ in its recurrent classes. If one does
           not exist, Return: $\mu^k$ with “not optimal”
10:    end if
11:    Set $k \leftarrow k+1$
12: until $\mu^k$ with gain-bias pair satisfying (27) and Return: $\mu^k$
    with “optimal”
```

Proposition V.5. *Given $\mathcal{C}$, Alg. 1 terminates in a finite number of iterations. If it returns policy $\mu_{\circlearrowright\mathcal{C}}$ with “optimal”, then $\mu_{\circlearrowright\mathcal{C}}$ satisfies the optimality conditions in Prop. V.4. If $\mathcal{C}$ is unichain (i.e., each stationary policy of $\mathcal{C}$ is unichain), then Alg. 1 is guaranteed to return the optimal policy $\mu^\star_{\circlearrowright\mathcal{C}}$.*

The unichain condition in Prop. V.5 is sufficient, but not necessary, *i.e.,* if $\mathcal{C}$ is not unichain, Alg. 1 may still return with “optimal”. If we obtain the optimal policy for each $\mathcal{C} \in \mathcal{A}$, then we use (26) to obtain the optimal solution for Prob. III.1. If for some $\mathcal{C}$, Alg. 1 returns “not optimal”, then the policy returned by Alg. 1 is only sub-optimal. We can then apply this algorithm to each AMEC in $\mathcal{A}$ and use (26) to obtain a sub-optimal solution for Prob. III.1. Note that similar to policy iteration algorithms for ACPS problems, either the gain or the bias strictly decreases every time when μ is updated, so policy μ is improved in each iteration. In both cases, the satisfaction constraint is always enforced.

VI. Case study

The algorithmic framework developed in this paper is implemented in MATLAB, and here we provide an example as a case study. Consider the MDP $\mathcal{M}$ shown in Fig. 1, which can be viewed as the dynamics of a robot navigating in an environment with the set of atomic propositions $\{\texttt{pickup}, \texttt{dropoff}\}$. In practice, this MDP can be obtained via an abstraction process (see *e.g.,* [20]) from the environment, where its probabilities of transitions can be obtained from experimental data or accurate simulations.

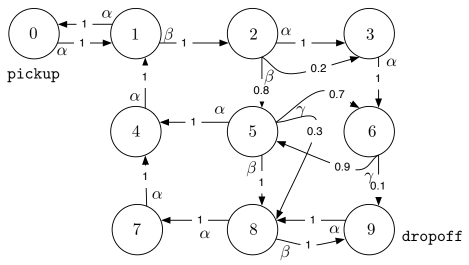

Fig. 1. MDP capturing a robot navigating in an environment. $\{\alpha, \beta, \gamma\}$ is the set of controls. The cost of applying α, β, γ at a state where the control is available is $5, 10, 1$, respectively. (*e.g.,* $g(i, \alpha) = 5$ if $\alpha \in U(i)$)

The goal of the robot is to continuously perform a pickup-delivery task. The robot is required to pick up items at the state marked by pickup (see Fig. 1), and drop them off at the state marked by dropoff. It is then required to go back to pickup and this process is repeated. This task can be written as the following LTL formula:

$$\phi = \Box\Diamond\texttt{pickup} \wedge \Box(\texttt{pickup} \Rightarrow \bigcirc(\neg\texttt{pickup}\,\mathcal{U}\texttt{dropoff})).$$

The first part of ϕ, $\Box\Diamond\texttt{pickup}$, enforces that the robot repeatedly pick up items. The remaining part of ϕ ensures that new items cannot be picked up until the current items are dropped off. We denote pickup as the optimizing proposition, and the goal is to find a policy that satisfies ϕ with probability 1 and minimizes the expected cost in between visiting the pickup state (*i.e.,* we aim to minimize the expected cost in between picking up items).

We generated the DRA $\mathcal{R}_\phi$ using the ltl2dstar tool [17] with 13 states and 1 pair $(L, K) \in F$. The product MDP $\mathcal{P}$ after removing unreachable states contains 31 states (note that $\mathcal{P}$ has 130 states without removing unreachable states). There is one AMEC $\mathcal{C}$ corresponding to the only pair in $F_\mathcal{P}$ and it contains 20 states. We tested Alg. 1 with a number of different initial policies and Alg. 1 produced the optimal policy within 2 or 3 policy updates in each case (note that $\mathcal{C}$ is not unichain). For one initial policy, the ACPC was initially 330 at each state of $\mathcal{C}$, and it was reduced to 62.4 at each state when the optimal policy was found. The optimal policy $\mu^\star_\mathcal{C}|_\mathcal{M}$ is as follows:

State	0	1	2	3	4	5	6	7	8	9
After pickup	α	β	α	α	α	γ	γ	α	β	α
After dropoff	α	α	α	α	α	α	γ	α	α	α

The first row of the above table shows the optimal policy after pick-up but before drop-off and the second row shows the optimal policy after drop-off and before another pick-up.

VII. Conclusions

We developed a method to automatically generate a control policy for a Markov Decision Process (MDP), in order to satisfy specifications given as Linear Temporal Logic formulas. The control policy satisfies the given specification almost surely, and it optimizes the average cost between satisfying instances of an "optimizing proposition", under some restrictions. The problem is motivated by robotic applications requiring some persistent tasks to be performed.

For future work, we aim for completeness and remove the restrictions of this paper in finding the optimal policy (if one exists). Alternatively, we aim to characterize the class of LTL formulas for which our algorithm is complete and always return the optimal policy.

References

[1] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *Proc ICRA*, Rome, Italy, 2007, pp. 3116–3121.

[2] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc CDC*, Shanghai, China, 2009, pp. 2222–2229.

[3] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *Proc CDC*, Paradise Island, Bahamas, 2004, pp. 153–158.

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

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

[6] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, Charleston, SC, 2006, pp. 364–380.

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

[8] M.Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Hybrid Systems: Computation and Control*, ser. Lect. Notes Comp. Science, M. Egerstedt and B. Mishra, Eds. Springer Verlag, 2008, pp. 287–300.

[9] L. De Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.

[10] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[11] M. Vardi, "Probabilistic linear-time model checking: An overview of the automata-theoretic approach," *Formal Methods for Real-Time and Probabilistic Systems*, pp. 265–276, 1999.

[12] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *Proc IFAC World C*, Milan, Italy, Aug. 2011, to appear.

[13] C. Courcoubetis and M. Yannakakis, "Markov decision processes and regular events," *IEEE Trans Automatic Ctrl*, vol. 43, no. 10, pp. 1399–1418, 1998.

[14] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, "Controller synthesis for probabilistic systems," in *Proceedings of IFIP TCS'2004*. Kluwer, 2004.

[15] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," *Int J Robotic Research*, 2011, In press.

[16] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control under probabilistic temporal logic constraints," March 2011, available at http://arxiv.org/abs/1103.4342.

[17] J. Klein, "ltl2dstar - LTL to deterministic Streett and Rabin automata," http://www.ltl2dstar.de/, 2007, viewed September 2010.

[18] D. Bertsekas, *Dynamic programming and optimal control, vol. II*. Athena Scientific, 2007.

[19] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons, 1994.

[20] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *Proc ICRA*, Anchorage, AK, 2010, pp. 3227 – 3232.