Robust Control of Uncertain Markov Decision Processes with Temporal Logic Specifications

Eric M. Wolff, Ufuk Topcu, and Richard M. Murray

Abstract—**We present a method for designing robust controllers for dynamical systems with linear temporal logic specifications. We abstract the original system by a finite Markov Decision Process (MDP) that has transition probabilities in a specified uncertainty set. A robust control policy for the MDP is generated that maximizes the worst-case probability of satisfying the specification over all transition probabilities in the uncertainty set. To do this, we use a procedure from probabilistic model checking to combine the system model with an automaton representing the specification. This new MDP is then transformed into an equivalent form that satisfies assumptions for stochastic shortest path dynamic programming. A robust version of dynamic programming allows us to solve for a ϵ-suboptimal robust control policy with time complexity $O(\log 1/\epsilon)$ times that for the non-robust case. We then implement this control policy on the original dynamical system.**

I. Introduction

As the level of autonomous operation expected of robots, vehicles, and other cyberphysical systems increases, there is a growing need for formal methods that allow desired system properties to be precisely specified and allow performance to be automatically verified. As autonomous systems often operate over long time periods in uncertain environments, it is also important that system performance is robust to both environmental disturbances and modeling errors.

A promising approach for specifying and verifying system properties uses temporal logics such as linear temporal logic (LTL) to specify tasks. LTL provides a natural framework to specify desired properties such as response (if A, then B), liveness (always eventually A), safety (always not B), stability (eventually always A), and priority (first A, then B, then C). This framework requires a finite-state abstraction for continuous dynamical system, which is possible, in the simulation sense, for a wide range of systems [1].

We use a Markov Decision Process (MDP) as a finite-state abstraction of our original dynamical system because it provides a general framework for modeling the non-determinism and probabilistic behavior that is present in many real-world systems. MDPs are also amenable to formal verification techniques for temporal logic properties [4]. These techniques generate a control policy for the MDP that maximizes the probability of satisfying a given LTL specification. However, these techniques assume that the state transition probabilities of the MDP are known exactly, which is often unrealistic. We relax this assumption by allowing the transition probabilities of our MDP abstraction to lie in uncertainty sets. We then generate a control policy that maximizes the worst-case probability of satisfying a given LTL specification over all transition probabilities in the uncertainty set.

We argue that it is prudent to consider uncertainty in the MDP model. Real systems have unmodeled dynamics and parametric uncertainty that are not fully captured by mathematical models. So, even if exact finite-state abstraction techniques are available for a dynamical system model, the resulting MDP abstraction will only represent the real system to the extent that the dynamical system model does. Furthermore, when approximate abstraction techniques are used for a dynamical system model, the MDP abstraction will be a further approximation of the real system.

There is a large amount of work in addressing the robustness of control policies for MDPs with uncertain transition probabilities. Our approach most closely follows that of Nilim and El Ghaoui [24], which addresses a wide variety of uncertainty models. Other work includes [3], [25], [13].

On the temporal logic side, formal verification is well developed for discrete systems represented as MDPs with exact transition matrices [9], [4] and there exist powerful software tools [18]. While there has been recent work in verification of uncertain MDPs, these works almost exclusively use simple interval models of transition probability uncertainty [28], [7], [6], which our work includes as a special case.

The above references are primarily concerned with discrete models. Controllers that are guaranteed to satisfy a given temporal logic specification can be generated for a variety of non-stochastic hybrid systems [17], [16], [27]. Similar methods for non-stochastic systems have been implemented on stochastic systems [14], [19]. Recent work that explicitly designs controllers for stochastic systems [10], [20] does not consider robustness. Robustness of non-probabilistic discrete transition systems to disturbances is explored in [22].

There are two main contributions in our work. First of all, we create a robust control policy π that maximizes the worst-case probability of satisfying an LTL specification φ for a system represented as a finite labeled MDP $\mathcal{M}$ with transition matrices in an uncertainty set $\mathcal{P}$. A control policy π is a mapping from each MDP state to an allowable action. The set $\mathcal{P}$ can be non-convex and it includes interval uncertainty sets as a special case. This freedom allows more statistically accurate and less conservative results than interval uncertainty sets. Our second contribution is a method for generating an approximate finite-state MDP abstraction from a possibly nonlinear dynamical system given in state-space form.

Authors are with the Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA. The corresponding author is `ewolff@caltech.edu`

We now give an informal overview of our solution technique; preliminary definitions and the formal problem statement are found in Sections II and III respectively. We first abstract the dynamical system as a finite labeled MDP using controllers generated from a linear approximation of the system and Monte Carlo simulation (Section VI). In Section V we combine this MDP with an automaton representation of the LTL specification to form a product MDP that represents system trajectories that satisfy both the system dynamics and the specification. We then use a dynamic programming approach (developed in Section IV) to create a robust control policy that maximizes the worst-case probability of satisfying the specification over all transition matrices in the uncertainty set. This can be viewed as a game between the system and its environment, where the controller selects actions to maximize the probability of satisfying the specification, while the environment selects transition matrices to minimize the probability of satisfying the specification. Finally, we implement this robust control policy on the original dynamical system. We present an example of our approach in Section VII and conclude with suggestions for future work in Section VIII.

II. Preliminaries

We now present formal definitions for Markov Decision Processes (MDPs), which we use as finite-state abstractions for the original continuous dynamics, and linear temporal logic (LTL), which is the task specification language. Throughout, we will assume that equality and inequality is component-wise for vectors and matrices. Also, $\mathbf{1}$ denotes a vector of ones of appropriate dimension.

A. Markov Decision Processes

MDPs are useful models for a wide range of systems as they provide a unified framework for systems with non-deterministic and stochastic aspects [5].

Definition 1 (Labeled finite MDP)**.** We specify a *labeled finite MDP*, $\mathcal{M}$, by the tuple $\mathcal{M} = (S, A, P, s_0, AP, L)$, where S is a finite set of states, A is a finite set of actions, $P : S \times A \times S \to [0,1]$ is the transition probability function, s_0 is the initial state, AP is a finite set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function. Let $A(s)$ denote the set of available actions at state s. $\sum_{s' \in S} P(s,a,s') = 1$ if $a \in A(s)$ and $P(s,a,s') = 0$ otherwise.

We assume, for notational convenience, that the available actions $A(s)$ are the same for every $s \in S$. We use P^a_{ij} as shorthand for the transition probability from state i to state j when using action a. We call $P^a \in \mathbb{R}^{n \times n}$ a *transition matrix*, where the (i,j)-th entry of P^a is P^a_{ij}. Where it is clear from context, we refer to the row vector P^a_i as p.

To model uncertainty in state transitions, we specify uncertainty sets for the transition matrices of the MDP. We assume that an uncertainty set $\mathcal{P}^a$ corresponding to action $a \in A$ is a set of probability matrices.

Assumption 1. $\mathcal{P}^a$ can be factored as the Cartesian product of its rows, so its rows are uncorrelated. Formally, for every $a \in A$, $\mathcal{P}^a = \mathcal{P}^a_1 \times \ldots \times \mathcal{P}^a_n$ where each $\mathcal{P}^a_i$ is a subset of the probability simplex in $\mathbb{R}^n$. We follow Nilim and El Ghaoui and refer to these as *rectangular* uncertainty sets [24].

Let F^a denote the nominal transition matrix for action a.

Assumption 2. $F^a_{ij} = 0$ if and only if $P^a_{ij} = 0$ for all $P^a \in \mathcal{P}^a$.

Intuitively, this means that if a nominal transition is zero (non-zero), then it is zero (non-zero) for all transition matrices in the uncertainty set. This must be enforced as adding or removing a transition to an unsafe set may result in a completely different satisfaction probability for properties over infinite time.

Definition 2. A *control policy* for an MDP $\mathcal{M}$ is a sequence $\pi = \{\mu_0, \mu_1, \ldots\}$, where each $\mu_k : S \to A$ such that $\mu_k(s) \in A(s)$. We call a policy *stationary* if it is of the form $\pi = \{\mu, \mu, \ldots\}$. We let Π be the set of all control policies and Π_s be the set of all stationary control policies.

Definition 3. An *environment policy* for an MDP $\mathcal{M}$ is a sequence of transition matrices $\tau = (P^a_k)_{a \in A, k=1,2,\ldots}$. We call an environment policy *stationary* if it is of the form $\tau = (P^a)_{a \in A}$. We let $\mathcal{T}$ be the set of all environment policies and $\mathcal{T}_s$ be the set of all stationary environment policies.

We now associate a cost with each state in $\mathcal{M}$ through a function $c(s,a) : S \times A \to \mathbb{R}$. This cost is incurred at every stage k and the control policy tries to minimize the total expected cost over the horizon of length N. The total expected cost of an infinite-horizon problem starting from state s when using control policy π under environment policy τ is

$$J^{\pi\tau}(s) := \lim_{N \to \infty} \mathbb{E}_{\pi\tau}\left[\sum_{k=0}^{N-1} c(s_k, \mu_k(s_k)) \big| s_0 = s\right], \qquad (1)$$

where the expectation $\mathbb{E}_{\pi\tau}$ depends on both the control and environment policies.

The optimal worst-case total expected cost starting from state s is

$$J^*(s) := \min_{\pi \in \Pi} \max_{\tau \in \mathcal{T}} J^{\pi\tau}(s). \qquad (2)$$

The optimal worst-case control and environment policies are denoted π^* and τ^*. We call π^* a *robust control policy*.

B. Linear Temporal Logic

We use linear temporal logic (LTL) to specify the desired system behavior. We will only touch on the key aspects of LTL for our problem; we defer the reader to [4] for details.

We describe an *execution* of a system by an infinite sequence of its states. Specifically, for a discrete-time system, its execution σ can be written as $\sigma = s_0 s_1 s_2 \ldots$ where $s_t \in S$ is the state of the system at time t.

Definition 4. An *atomic proposition* is a statement that has a unique truth value (*True* or *False*). Let $s \in S$ be a state of the system and p be an atomic proposition. We write $s \Vdash p$ if p is *True* at the state s.

LTL is a powerful specification language for unambiguously and concisely expressing a wide range of properties of systems [12]. It is built up from (a) a set of atomic propositions, (b) the logic connectives: negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$) and material implication ($\Longrightarrow$), and (c) the temporal modal operators: next ($\bigcirc$), always ($\square$), eventually ($\diamondsuit$) and until ($\mathcal{U}$).

An LTL formula is defined inductively as follows: (1) any atomic proposition p is an LTL formula; and (2) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi\,\mathcal{U}\,\psi$ are also LTL formulas. Other operators can be defined as follows: $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \Longrightarrow \psi := \neg\varphi \vee \psi$, $\diamondsuit\varphi := \mathit{True}\,\mathcal{U}\,\varphi$, and $\square\varphi := \neg\diamondsuit\neg\varphi$.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution $\sigma = s_0 s_1 s_2 \ldots$ and an LTL formula φ, we write $s_i \vDash \varphi$ if φ holds at position $i \geq 0$ of σ.

The semantics of LTL is defined inductively as follows: (a) For an atomic proposition p, $s_i \vDash p$ if and only if (iff) $s_i \Vdash p$; (b) $s_i \vDash \neg\varphi$ iff $s_i \nvDash \varphi$; (c) $s_i \vDash \varphi \vee \psi$ iff $s_i \vDash \varphi$ or $s_i \vDash \psi$; (d) $s_i \vDash \bigcirc\varphi$ iff $s_{i+1} \vDash \varphi$; and (e) $s_i \vDash \varphi\,\mathcal{U}\,\psi$ iff there exists $j \geq i$ such that $s_j \vDash \psi$ and $\forall k \in [i,j), s_k \vDash \varphi$. Based on this definition, $\bigcirc\varphi$ holds at position s_i iff φ holds at the next state s_{i+1}, $\square\varphi$ holds at position i iff φ holds at every position in σ starting at position i, and $\diamondsuit\varphi$ holds at position i iff φ holds at some position $j \geq i$ in σ.

Definition 5. An execution $\sigma = s_0 s_1 s_2 \ldots$ *satisfies* φ, denoted by $\sigma \vDash \varphi$, if $s_0 \vDash \varphi$.

Remark 1. Properties typically studied in the control and hybrid systems domains are safety and stability. LTL generalizes this by also allowing properties such as guarantee, obligation, liveness, and response.

We next define a deterministic Rabin automaton, which provides an automaton representation for any LTL formula. We use this instead of a nondeterministic Buchi automaton because nondeterminism can change the probability measure over the product MDP that we define in Section V. We cannot use a determinstic Buchi automaton, as these cannot express all LTL formulae [4].

Definition 6. A deterministic Rabin automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where Q is a finite set of states, Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and accepting state pairs $Acc \subseteq 2^Q \times 2^Q$.

Let Σ^ω be the set of infinite words over Σ. A run for $\sigma = \mathcal{A}_0\mathcal{A}_1\mathcal{A}_2 \ldots \in \Sigma^\omega$ denotes an infinite sequence $q_0 q_1 q_2 \ldots$ of states in $\mathcal{A}$ such that $q_{i+1} \in \delta(q_i, \mathcal{A}_i)$ for $i \geq 0$. The run $q_0 q_1 q_2 \ldots$ is accepting if there exists a pair $(L, K) \in Acc$ and an $n \geq 0$, such that for all $m \geq n$ we have $q_k \notin L$ and there exist infinitely many k such that $q_k \in K$.

Intuitively, a run is accepted by a deterministic Rabin automaton if the set of states L is visited finitely often and the set K is visited infinitely often.

III. Problem Statement

We now provide a formal statement of the problem and an overview of our approach.

The system evolves with the discrete-time dynamics

$$x_{t+1} = f(x_t, u_t, w_t) \quad \text{for} \quad t = 0, 1, \ldots, \tag{3}$$

where $x_t \in \mathbb{R}^l$ is the system state, $u_t \in \mathbb{R}^m$ is the control input, and $w_t \in \mathbb{R}^p$ is a stochastic disturbance input.

We assume that relevant system properties are specified by a given LTL formula φ over a finite set AP of atomic propositions. Let $\mathcal{X} \subset \mathbb{R}^l$ be partitioned into a finite set $\mathcal{R}$ of regions. Each region $r \in \mathcal{R}$ has an associated set of *True* atomic propositions associated with it, which we will informally call a *label*. Let $R : \mathcal{X} \to \mathcal{R}$ be a surjective mapping that associates each system state $x \in \mathcal{X}$ with a region $r \in \mathcal{R}$. Furthermore, we associate each region with a labeling function $L : \mathcal{R} \to 2^{AP}$. We assume a bijection between regions $r \in \mathcal{R}$ and MDP states $s \in S$, hence we use the same notation for this labeling function as in Definition 1. Since every system state $x \in \mathcal{X}$ in a given region has the same label, this is a *proposition-preserving* partition. Finally, let $\mathcal{S} : 2^{AP} \to \mathcal{X}$ which maps labels to system states.

Given a control policy $\pi = \{\mu_0, \mu_1, \ldots\}$ the MDP abstraction of the dynamical system (3), we can induce a control policy π^*_{proj} on (3) by using action $\mu_k(R(x))$ for the system state $x \in \mathcal{X}$ at time $k = 0, 1, \ldots$. This simply maps the system state to the corresponding region with R, and thus a unique MDP state due to the assumed bijection.

Problem 1. Given a labeled finite MDP $\mathcal{M}$ with transition matrices in an uncertainty set $\mathcal{P}^a$ for each action $a \in A$ and an LTL specification φ over AP, create a robust control policy π^* for $\mathcal{M}$. Furthermore, if $\mathcal{M}$ is an abstraction of a dynamical system (3), create the induced control policy π^*_{proj} for (3).

We present a method for abstracting the original system as a finite MDP $\mathcal{M}$ with transition matrix uncertainty sets in Section VI. We then solve Problem 1 by first creating the product MDP $\mathcal{M}_p$ which contains system trajectories that satisfy both $\mathcal{M}$ and the deterministic Rabin automaton representing the given LTL specification. We modify $\mathcal{M}_p$ so that all policies for it are proper, and thus it satisfies the stochastic shortest path assumptions. We then show that maximizing the worst-case probability of satisfying the specification is equivalent to a creating a control policy that maximizes the worst-case probability of reaching a certain set of states in $\mathcal{M}_p$. We solve for this policy using a robust variant of dynamic programming. Finally, we map the robust control policy back to $\mathcal{M}$ and the original dynamical system (3).

IV. Robust Dynamic Programming

In this section, we prove when a robust form of dynamic programming holds for MDPs with uncertain transition matrices.

A. Dynamic Programming

We consider a MDP $\mathcal{M}$ with a finite set of states $S = \{1, 2, \ldots, n, t\}$ and actions $a \in A(s)$ for all $s \in S$. We assume that t is a special terminal state, which is absorbing ($P^a_{tt} = 1$) and incurs zero cost ($c(t) = 0$) for all $a \in A(t)$ and all $P \in \mathcal{P}^a_t$ [5]. It follows that the total expected cost for t, $J(t) = 0$ for all control and environment policies.

We require policies to be proper, i.e. they almost surely reach the terminal state t for all transition matrices in the uncertainty set [5].

Definition 7. A stationary control policy μ is *proper* if, under that policy, there is positive probability that the terminal state will be reached after at most n stages, regardless of the initial state and transition matrices, that is, if

$$\rho_{\mu\tau} := \max_{s=1,\ldots,n} \max_{P \in \mathcal{P}^{\mu(s)}} P(s_n \neq t | s_0 = s, \mu) < 1. \tag{4}$$

Assumption 3. All stationary control policies are proper.

Remark 2. This assumption intuitively means that the terminal state will eventually be reached under any stationary policy. This will allow us to make statements regarding convergence rates. While this is usually a rather strong condition, these assumptions are not restrictive for Problem 1, as shown in Section V.

In preparation for the main result of this section, we give the following classical definition and theorem [23].

Definition 8. Let (M, d) be a metric space and $f : M \to M$. We say that f is a *contraction* if there is a real number β, $0 \leq \beta < 1$, such that

$$d(f(x), f(y)) \leq \beta d(x, y)$$

for all x and y in M.

Theorem 1 (Contraction Mapping Theorem). *Let (M, d) be a complete metric space and let $f : M \to M$ be a contraction. Then there exists a unique point x^* in M such that*

$$f(x^*) = x^*.$$

Additionally, if x is any point in M, then

$$\lim_{k\to\infty} f^k(x) = x^*,$$

where f^k is the composition of f with itself k times.

We now define mappings that play an important role in the rest of this section. We loosely follow the notation of [5]. The scalar $J(s)$ corresponds to the total expected cost to go when starting at state $s \in S$. The shorthand $J \in \mathbb{R}^n$ represents the cost vector for all $s \in S \backslash t$. Since the cost is always zero at the terminal state t, we do not include it. The T and $T_{\mu\tau}$ operators are mappings from $\mathbb{R}^n$ to $\mathbb{R}^n$. For each state $s \in S \backslash t$, define the s-th component of TJ and $T_{\mu\tau}J$ respectively as

$$(TJ)(s) := \min_{a \in A(s)} [c(s, a) + \max_{p \in \mathcal{P}^a_s} p^T J], \tag{5}$$

$$(T_{\mu\tau}J)(s) := c(s, \mu(s)) + p^T J. \tag{6}$$

In the following two lemmas, we show that these mappings are monotonic and contractive. We prove these for (5); the proofs for (6) follow by limiting the actions and transition probabilities at each state s to $\mu(s)$ and P^a_s respectively. T^k is the composition of T with itself k times.

Lemma 1 (Monotonicity). *For any vectors $u, v \in \mathbb{R}^n$, such that $u \leq v$, we have that $T^k u \leq T^k v$ for $k = 1, 2, \ldots$.*

Proof: Immediate from (5) and definition of $\mathcal{P}^a_s$. ∎

Definition 9. The *weighted maximum norm* $\| \cdot \|_w$ of a vector $u \in \mathbb{R}^n$ is defined by

$$\| u \|_w = \max_{i=1,\ldots,n} \frac{|u(i)|}{w(i)}$$

where vector $w \in \mathbb{R}^n$ and $w > 0$.

Lemma 2 (Contraction). *If all stationary control policies are proper, then there exists a vector $w > 0$ and a scalar $\gamma \in [0, 1)$ such that $\| Tu - Tv \|_w \leq \gamma \| u - v \|_w$ for all $u, v \in \mathbb{R}^n$.*

Sketch of proof: The proof of Lemma 2 closely follows that in [5] (Vol. II, Section 2.4) where the environment policy is fixed. More specifically, the proof in [5] is modified to allow maximization over environment policies. This modification holds due to Assumption 3 and Lemma 1. Details are available in the Appendix. ∎

We now prove the main result of this section. We remind the reader that the vector $J^* \in \mathbb{R}^n$, defined in (2), is the optimal worst-case total expected cost starting from state $s \in S$.

Theorem 2 (Robust Dynamic Programming). *Under the assumption that all stationary control policies μ are proper for a finite MDP $\mathcal{M}$ with transition matrices in the uncertainty set $\mathcal{P}^a$ for $a \in A$, the following statements hold.*

(a) The optimal worst-case cost vector J^* is the unique fixed-point of T,

$$J^* = TJ^*. \tag{7}$$

(b) The optimal worst-case cost vector J^* is given by,

$$J^* = \lim_{k\to\infty} T^k J, \tag{8}$$

for all $J \in \mathbb{R}^n$. This limit is unique.

(c) A stationary control policy μ and a stationary environment policy τ are optimal if and only if

$$T_{\mu\tau} J^* = TJ^*. \tag{9}$$

Proof: Parts (a) and (b) follow immediately from Theorem 1 and Lemmas 1 and 2.

Part (c): First assume that $T_{\mu\tau}J^* = TJ^*$. Thus $T_{\mu\tau}J^* = TJ^* = J^*$ from (7) and $J^{\mu\tau} = J^*$ from the uniqueness of the fixed-point. Thus, μ and τ are optimal policies. Now assume that μ and τ are optimal policies so that $J^{\mu\tau} = J^*$. We then have that $T_{\mu\tau}J^* = T_{\mu\tau}J^{\mu\tau} = J^{\mu\tau} = J^*$. ∎

Corollary 1. *Given the optimal worst-case cost vector J^*, the optimal control actions a^* satisfy*

$$a^*(s) \in \arg\min_{a \in A(i)} [c(s, a) + \max_{p \in \mathcal{P}^a_s} p^T J^*], \quad s \in S. \tag{10}$$

and, with some abuse of notation, the optimal transition vectors (for the environment) are

$$P_s^{*a} \in \arg\max_{p \in \mathcal{P}_s^a} p^T J^*, \quad s \in S, a \in A(s). \tag{11}$$

Proof: Follows from Part (c) in Theorem 2 and (5). ■

To recap, we showed that T is monotone and a contraction with respect to a weighted max norm. This allowed us to prove in Theorem 2 that T has a unique fixed-point that can be found by an iterative procedure (often called *value iteration*). We further gave conditions on the optimality of stationary policies and showed how to determine optimal actions for the system and the environment.

B. Uncertainty Set Representations

Refering back to the T operator (5), we see that it is composed of two nested optimization problems–the outer problem for the system and the inner problem for the environment. To be clear, the environment optimization problem for a given state $s \in S$ and control action $a \in A(s)$ refers to

$$\max_{p \in \mathcal{P}_s^a} p^T J. \tag{12}$$

The tractability of this optimization problem depends on the structure of the uncertainty set $\mathcal{P}_s^a$. In the remainder of this section, we investiate interval and likelihood uncertainty sets, as these are both statistically meaningful and computationally efficient. Due to lack of space, we do not discuss maximum a priori, entropy, scenario, or ellipsoidal uncertainty models, even though these are included in this framework. The interested reader should refer to Nilim and El Ghaoui for details [24].

1) Interval Models: A common description of uncertainty for transition matrices corresponding to action $a \in A$ is by intervals

$$\mathcal{P}^a = \{P^a \,|\, \underline{P}^a \leq P^a \leq \overline{P}^a,\ P^a\mathbf{1} = 1\}, \tag{13}$$

where $\underline{P}^a$ and $\overline{P}^a$ are nonnegative matrices $\underline{P}^a \leq \overline{P}^a$. This representation is motivated by statistical estimates of confidence intervals on the individual components of the transition matrix. The environmental optimization problem can be solved in $O(n\log(n))$ time using a bisection method [24].

2) Likelihood Models: The likelihood uncertainty model is motivated by determining the transition probabilities between states through experiments. We denote the experimentally measured transition probability matrix corresponding to action a by F^a and the optimal log-likelihood by $\beta_{\max}$.

Uncertainty in the transition matrix for each action $a \in A$ is described by the *likelihood region* [21]

$$\mathcal{P}^a = \{P^a \in \mathbb{R}^{n\times n} | P^a \geq 0, P^a\mathbf{1} = \mathbf{1}, \sum_{i,j} F_{ij}^a \log P_{ij}^a \geq \beta^a\}, \tag{14}$$

where $\beta^a < \beta^a_{\max}$ and can be estimated for a desired confidence level by using a large sample Gaussian approximation [24]. As described in Assumption 2, we enforce that $F_{ij}^a = 0$ if and only if $P_{ij}^a = 0$ for all $i, j \in S$ and all $a \in A$.

Since the likelihood region above is not rectangular (contrary to Assumption 1), it must be approximated by projection onto each row of the transition matrix. Even with the approximation, likelihood regions are less conservative uncertainty representations than intervals, which arise from further projection onto the row's components. A bisection algorithm can approximate the environment optimization problem to within an accuracy δ in $O(\log(J_{\max}/\delta))$ time, where $J_{\max}$ is the maximum value of the cost vector, J [24].

V. Creating the Robust Control Policy

We now solve Problem 1 by creating a robust control policy. First, we form the product MDP $\mathcal{M}_p$ which allows behaviors that satisfy both the system MDP $\mathcal{M}$ and the LTL specification φ. We show that $\mathcal{M}_p$ can be transformed into an equivalent form $\mathcal{M}_{ssp}$ where all stationary control policies are proper. Next, we use the robust dynamic programming developed in Section IV to find a control policy that maximizes the probability of satisfying φ. Finally, we project this control policy to a policy for $\mathcal{M}$.

A. Forming the product MDP

The product MDP $\mathcal{M}_p$ restricts behaviors to those that satisfy both the system transitions and the determinstic Rabin automaton $\mathcal{A}$ representing the LTL specification.

Definition 10. For labeled finite MDP $\mathcal{M} = (S, A, P, s_0, AP, L)$ and deterministic Rabin automaton $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$, the *product MDP* $\mathcal{M}_p = (S_p, A, P_p, s_{0p}, Q, L_p)$ with

- $S_p = S \times Q$,
- $P_p((s,q),\alpha,(s',q')) = \begin{cases} P(s,\alpha,s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise,} \end{cases}$
- $s_{0p} = (s_0, q)$ such that $q = \delta(q_0, L(s_0))$,
- $L_p((s,q)) = \{q\}$.

The accepting product state pairs $Acc_p = \{(L_1^p, K_1^p), \ldots, (L_k^p, K_k^p)\}$ are lifted directly from Acc. Formally, for every $(L_i, K_i) \in Acc$, state $(s,q) \in S_p$ is in L_i^p if $q \in L_i$, and $(s,q) \in K_i^p$ if $q \in K_i$.

There is a one-to-one correspondence between the paths on $\mathcal{M}_p$ and $\mathcal{M}$, which induces a one-to-one correspondence for policies on $\mathcal{M}_p$ and $\mathcal{M}$. So, given a policy $\pi^p = \{\mu_0^p, \mu_1^p, \ldots\}$ on $\mathcal{M}_p$, we can induce a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ on $\mathcal{M}$ by setting $\mu_i(s_i) = \mu_i^p((s_i, q_i))$ for every stage $i = 0, 1, \ldots$. This is always valid since $\mathcal{M}_p$ and $\mathcal{M}$ have the same action set A. If π^p is stationary, π is guaranteed to be finite-memory, but not necessarily stationary [4].

B. Reachability in product MDP

We now show how to use the product MDP $\mathcal{M}_p$ to determine a robust control policy that maximizes the worst-case probability that a given LTL specification is satisfied. Given a control and environment policy, the probability of satisfying an LTL formula is equivalent to the probability of reaching a certain set of states in $\mathcal{M}_p$, which are *accepting maximal end components* [4]. We call this probability the *reachability probability*. Informally, accepting maximal end components are sets of states that the system can remain in forever and where the acceptance condition of the determinsitic Rabin automaton is satisfied.

Definition 11. A *sub-MDP* of a MDP is a pair of states and action sets (C, D) where: (1) $\emptyset \neq C \subseteq S$ and $D : C \to 2^A$ is a function such that $\emptyset \neq D(s) \subseteq A(s)$ for all states $s \in C$ and (2) $s \in C$ and $a \in D(s)$ implies $Post(s,a) = \{t \in S | P^a_{st} > 0\} \subseteq C$.

Definition 12. An *end component* is a sub-MDP (C, D) such that the digraph $G_{(C,D)}$ induced by (C, D) is strongly connected.

An end component (C, D) is *maximal* if there is no end component (C', D') such that $(C, D) \neq (C', D')$ and $C \subseteq C'$ and $D(s) \subseteq D'(s)$ for all $s \in C$. Furthermore, (C, D) is *accepting* for the deterministic Rabin automaton $\mathcal{A}$ if for some $(L, K) \in Acc$, $L \notin C$ and $K \in C$.

After computing the accepting maximal end components of $\mathcal{M}_p$, we need to determine a control policy that maximizes the worst-case probability of reaching an accepting maximal end component from our initial state. Without considering transition probability uncertainty, this policy can now be computed using either linear or dynamic programming methods [4]. We need to use the robust dynamic programming approach from Section IV. Note that this approach does not directly apply to $\mathcal{M}_p$, as not all stationary control policies on $\mathcal{M}_p$ are proper. We can transform $\mathcal{M}_p$ into an equivalent MDP where all stationary policies are proper, as shown in the next subsection.

C. Transformation to Stochastic Shortest Path

We now show how to transform the product MDP $\mathcal{M}_p$ into an equivalent form $\mathcal{M}_{ssp}$ where all stationary control policies μ are proper (cf. Assumption 3). It is important to note that $\mathcal{M}_p$ and $\mathcal{M}_{ssp}$ are equivalent only in terms of the probability of reaching an accepting maximal end component—both the states and the transition probabilities will likely change.

In the remainder of this section, we use the simplified notation $\mathcal{M}_p = (S, A, P)$ to describe the states, actions, and transition matrices of the product MDP. We refer to the state of $\mathcal{M}_p$ as s instead of (s, q) when clear from context.

We first partition the states S of $\mathcal{M}_p$ into three disjoint sets. We let B be the union of all accepting maximal end components in $\mathcal{M}_p$. By definition, every state $s \in B$ has a maximum reachability probability of 1. We define the set of states that have zero probability of reaching B by S_0. These can be found efficiently by graph algorithms [4]. Finally, we define the set $S_r = S - (B \cup S_0)$ as all of states not in an accepting maximal end component with non-zero maximum reachability probability. It is easy to see that these three sets form a partition of S.

Algorithm 1 Appending the terminal state
```
Require: $\mathcal{M}_p = (S, A, P)$ and $S_r, S_0, B$
  $S := S \cup \{t\}$ and $A(t) := \{u\}$ and $c(t,u) := 0$
  $A(s) := \{u\}$ and $P^u_{st} := 1$ for all $s \in B \cup S_0$
```

In Algorithm 1, we augment S with a terminal state t which is absorbing and incurs zero cost (cf. Section IV).

Remark 3. Algorithm 1 does not change the probabilty of reaching an accepting maximal end component for any state $s \in S$ under any control and environment policies.

Algorithm 2 End component elimination (de Alfaro [9])
```
Require: MDP $\mathcal{M}_p = (S, A, P)$ and $S_r, S_0, B$
Ensure: MDP $\mathcal{M}_{ssp} = (\hat{S}, \hat{A}, \hat{P})$
  $\{(C_1, D_1), \dots, (C_k, D_k)\}$ max end components in $S_r$
  $\hat{S}_0 := S_0$ and $\hat{B} := B$
  $\hat{S} := S \cup \{\hat{s}_1, \dots, \hat{s}_k\} - \cup_{i=1}^k C_i$
  $\hat{S}_r := S_r \cup \{\hat{s}_1, \dots, \hat{s}_k\} - \cup_{i=1}^k C_i$
  $\hat{A}(s) := \{(s,a) \mid a \in A(s)\}$ for $s \in S - \cup_{i=1}^k C_i$
  $\hat{A}(\hat{s}_i) := \{(s,a) \mid s \in C_i \wedge a \in A(s) - D(s)\}$ for $1 \le i \le k$
  For $s \in \hat{S}, t \in S - \cup_{i=1}^k C_i$ and $(u,a) \in \hat{A}(s)$, $\hat{P}^{(u,a)}_{st} := P^a_{ut}$
  and $\hat{P}^{(u,a)}_{s\hat{s}_i} := \sum_{t \in C_i} P^a_{ut}$
```

Algorithm 2 eliminates the maximal end components in S_r and replaces them with new states. This procedure is from Algorithm 3.3 of [9], where it is also proven (Theorem 3.8 in [9]) that the reachability probabilities are unchanged by this procedure. The intuition behind this result is that one can move between any two states r and s in a maximal end component in S_r with probability one and zero cost.

After applying Algorithms 1 and 2, we call the resulting MDP $\mathcal{M}_{ssp}$. We note that $\hat{S}_r$, $\hat{B}$, and $\hat{S}_0$ form a disjoint partition of $\hat{S}$. All stationary control policies for $\mathcal{M}_{ssp}$ are proper, i.e., they will almost surely reach the terminal state t.

Theorem 3. *All stationary control policies for $\mathcal{M}_{ssp}$ are proper.*

Proof: By contradiction. Suppose instead that there exists a stationary control policy μ such that the system starting in state $s_0 \in \hat{S}_r$ never reaches the terminal state t (i.e. there does not exist an integer k such that $s_k = t$). It should be clear that we only need to look at $s_0 \in \hat{S}_r$, as all $s \in \hat{B} \cup \hat{S}_0$ deterministically transition to t. This implies that there does not exist a transition between any states $r \in \hat{S}_r$ and $s \in \hat{B} \cup \hat{S}_0$. Therefore, we have that under μ, there exists a set $U \in \hat{S}_r$ such that if state $s_k \in U$ for some finite integer k, then $s_k \in U$ for all k. This means that U must be an end component in $\hat{S}_r$, which is the contradiction. ∎

Thus, $\mathcal{M}_{ssp}$ is equivalent in terms of reachability probabilities to the original product MDP $\mathcal{M}_p$. Furthermore, all stationary control policies are proper.

D. Computing the optimal control policy

We now perform robust value iteration as described in Section IV on the transformed product MDP $\mathcal{M}_{ssp}$. This method applies because M_{ssp} has been constructed so that all stationary policies are proper.

As we formulated the dynamic programming approach in terms of total expected cost minimization, we define the total expected cost as the negative of the reachability probabilty, which is equivalent to the probability of satisfying the LTL formula. We refer to cost and probability interchangably in the remainder. Thus, for all $a \in \hat{A}$, the appropriate costs are $c(s,a) = -1$ for all $s \in \hat{B}$ and $c(s,a) = 0$ for all $s \in \hat{S}_0$. For the remaining states, $s \in \hat{S}_r$, we initialize the costs arbitrarily in $[-1, 0]$ and compute the optimal worst-case cost vector using the iteration presented in Theorem 2. The resulting cost vector is $J^*_{ssp} \in \mathbb{R}^m$, where m is the number of states in $\mathcal{M}_{ssp}$.

We first determine the cost vector $J^*_p \in \mathbb{R}^n$ for $\mathcal{M}_p$ from $J^*_{ssp} \in \mathbb{R}^m$. For $s_p \in S_p$, we determine the corresponding state $s_{ssp} \in \hat{S}$ and let $J^*_p(s_p) = J^*_{ssp}(s_{ssp})$. This mapping is surjective, as there is at least one s_p for each s_{ssp}.

Now that we have the optimal worst-case cost vector J^*_p for the original product MDP $\mathcal{M}_p$, we need to determine the optimal actions $a^*(s) \in A(s)$ for each $s \in S_r$. We do not consider actions for states in $S_0 \cup B$ at this time. However, we cannot simply use the approach for selecting actions given by (10), because not all stationary control policies on $\mathcal{M}_p$ are proper. For states in a maximal end component in S_r, there may be multiple actions that satisfy (10). Arbitrarily selecting actions can lead to situations where the stationary control policy stays in the maximal end component forever and thus never satisfies the specification. We avoid this situation by only selecting an action if it is both optimal (i.e., satisfies (10)) and it has a non-zero probability of transitioning to a state that is not in a maximal end component in S_r. Algorithm 3 selects the action with the highest probability of transitioning to a state not in a maximal end component in S_r.

Algorithm 3 Product MDP Control Policy

```
Require: $J^*_p \in \mathbb{R}^n$, $\mathcal{M}_p = (S, A, P)$, $S_r$, $S_0$, $B$
  $visited := S_0 \cup B$
  $possAct(s) := \{a \in A(s) | (T_a J^*_p)(s) = J^*_p(s)\}$
  for $s \in S_r$ do
    if $|possAct(s)| = 1$ then
      $\mu(s) := possAct(s)$ and $visited := visited \cup \{s\}$
    end if
  end for
  while $visited \neq S$ do
    for $s \in S_r \backslash visited$ do
      $maxLeaveProb := 0$
      $leaveProb := \max_{a \in possAct(s)} \sum_{t \in visited} P^a_{st}$
      if $leaveProb > maxLeaveProb$ then
        $optAct := a$ and $optState := s$
      end if
    end for
    $\mu(s) := optAct$ and $visited := visited \cup \{optState\}$
  end while
```

Theorem 4. *Algorithm 3 returns a robust control policy μ that satisfies $J^{\mu\tau}_p = J^*_p$ for the worst-case environmental policy τ.*

Proof: For each state $s \in S_r$, we only need to consider actions $a \in A(s)$ that satisfy $(T_a J^*)(s) = J^*(s)$ as all other actions cannot be optimal with respect to the worst-case environmental policy τ. We call these *possible actions*. From the previous dynamic programming on $\mathcal{M}_{ssp}$ and the definition of end component, every state has at least one possible action. We say state $s \in visited$ if a possible action has been selected for it that also has a positive probability of leaving S_r. Thus, states in $visited$ are not in an end component in S_r. Initialize $visited = S_0 \cup B$. For every state with only one possible action, select that action and add the state to $visited$. For states with multiple possible actions, only select an action if it has a non-zero probability of reaching $visited$, and thus, leaving S_r. It is always possible to choose an action in this manner from the definition of S_r. Select actions this way until $visited = S$ and return the corresponding policy μ. By construction, μ satisfies $T_\mu J^* = J^*$ and is guaranteed to eventually exit S_r. ∎

The optimal control policy for satisfying the LTL specification φ consists of two parts: a stationary deterministic policy for reaching an accepting maximal end component, and a finite-memory deterministic policy for staying there. The former policy is given by Algorithm 3 and denoted μ_{reach}. The latter policy is a finite-memory controller π_B that selects actions in a round-robin fashion to ensure that the system stays inside the accepting maximal end component for all time and thus satisfies φ [4]. The overall optimal policy is $\pi^*_p = \mu_{reach}$ if $s \notin B$ and $\pi^*_p = \pi_B$ if $s \in B$. We induce an optimal policy π^* on $\mathcal{M}$ as described in Section V-A.

Remark 4. If there is another criteria that is important besides maximizing the probability of satisfying the specification, the control policy inside the AMEC can be modified accordingly. The only requirement is that it still ensure that the AMEC is never left and all states inside are visited infinitely often. An example of this approach is given in [11].

E. Complexity

We now detail the complexity of our approach for solving Problem 1. The number of states in the system MDP $\mathcal{M}$ is the number of labeled regions in $\mathcal{R}$ due to the bijection between MDP states and regions. The number of transitions for each state is given by the reachability between regions $r \in \mathcal{R}$, and is usually sparse for robot motion planning problems.

The size of the deterministic Rabin automaton $\mathcal{A}$ is, in the worst case, doubly-exponential in the length of the LTL formula [8]. Experimental work in [15] has shown that deterministic Rabin automaton sizes are often exponential or lower for many common types of LTL formulae. Also, there are fragments of LTL, which include all safety and guarantee properties, that generate a determinstic Rabin automaton whose size is singly-exponential in the length of the formula [2].

The size of the product MDP $\mathcal{M}_p$ is equal to the size of $\mathcal{M}$ times the size of $\mathcal{A}$. $\mathcal{M}_p$ has n states and m transitions. Maximal end components can be found in $O(n^2)$ time. Since T is a contraction, an ϵ-suboptimal control policy can be computed in $O(n^2 m \log(1/\epsilon))$ time without uncertainty sets [5] and $O(n^2 m \log(1/\epsilon)^2)$ when using likelihood transition uncertainty sets. Thus, the computational cost for incorporating robustness is $O(\log(1/\epsilon))$ times the non-robust case.

VI. Creation of MDP and Uncertainty Sets

We now discuss a simulation-based approach for creating a finite MDP abstraction $\mathcal{M}$ of a dynamical system (3). It should be noted that the main contribution of our work, the solution of Problem 1, does not depend on the method for creating $\mathcal{M}$ and its corresponding transition matrix uncertainty sets $\mathcal{P}$. Thus, if specific systems are more amenable to other abstraction methods, these can be used to create $\mathcal{M}$ and $\mathcal{P}$, after which our methods can compute a robust control policy. An approach using control primitives is given in [19].

Given a dynamical system (3) and a set $\mathcal{R}$ of labeled regions, where the labels correspond to sets of *True* atomic propositions that an LTL formula is defined over, we want to compute reachability for the system between all regions in $\mathcal{R}$. We say that for two regions $a, b \in \mathcal{R}$, region b is *reachable* from region a, if there exists a control sequence $u_{1:t}$ for every system state in a that evolves the system to some system state in region b in finite time.

Computing reachability is hard in general, so it is often necessary to consider a simplified model of (3) where reachability can be efficiently computed. Given candiate control sequences based on reachability analysis of the simplified system, one can use simulations to estimate the effect of these control actions on the full system dynamics and use these estimates of reachability to create an MDP abstraction. For every region, randomly sample N initial system states inside it and apply the candidate control sequences. Based on the average of all N simulated trajectories of the system (3) between regions i and j, create the estimated transition probability matrix F^a_{ij} entries. It is important to consider a transition between regions i and j successful only if the trajectory remains in the union of the two regions. This is because one cannot guarantee that a LTL specification is satisfied if the system visits other regions in between. Trajectories that visit multiple regions can transition to a special 'catch-all' state that has zero probability of satisfying the specification. The designer can tradeoff between the number of samples N and the size of the transition matrix uncertainty sets.

An uncertanty set for the transition matrices of $\mathcal{M}$ can be generated using the simulation data. Methods for determining these sets in a principled manner can be found in [24].

Remark 5. Even after simulating an arbitrary number of samples, we still can't be sure that the estimated transition matrices are exact, as the system (3) might not accurately model the physical system. Thus, even if an abstraction method could generate an MDP from (3) that was exact, uncertainty sets should still be considered on the transition matrices.

The uncertanty set for the transition matrices of $\mathcal{M}$ can be generated using the results of the above simulation. Methods for determining these sets in a principled manner can be found in [24].

VII. Example

We demonstrate our robust control framework on a mobile robot, where there is uncertainty in the control inputs from stochastic disturbances caused by terrain variation.

Algorithm 4 Create MDP abstraction

```
Require: $N$, original (3) and simplified system, regions $\mathcal{R}$
Ensure: MDP $\mathcal{M} = (S, A, F, AP, L)$
  $S = \{r | r \in \mathcal{R}\}$ and determine $AP$ and $L$ from $\mathcal{R}$
  Let $r' \in reach(r)$ if $r'$ is reachable from $r$ for simplified
  system
  for $r \in \mathcal{R}$ do
    for $r' \in reach(r)$ do
      for $i = 1 \to N$ do
        $x \leftarrow$ random system state in $r$
        $u_{1:T-1} \leftarrow getControl(x, r, r')$
        for $t = 1 \to T-1$ do
          $x_{t+1} \leftarrow f(x_t, u_t, w_t)$ {simulate full dynamics}
        end for
        $r'' \leftarrow R(x_T)$ $\{R : \mathcal{X} \to \mathcal{R}\}$
        $F(r, A_{r,r'}, r'') += \frac{1}{N}$ {where $F(i,u,j) = F^u_{ij}$}
      end for
    end for
  end for
```

We use the discrete-time form of the standard unicycle model for our system

$$\begin{aligned} x_{k+1} &= x_k + (v + d_v)\cos(\theta)\Delta t \\ y_{k+1} &= y_k + (v + d_v)\sin(\theta)\Delta t \\ \theta_{k+1} &= \theta_k + (\omega + d_\omega)\Delta t \end{aligned}$$

where $\Delta t = t_{k+1} - t_k$, v is the velocity input, ω is the angular velocity input, and $d = (d_v, d_\omega)^T$ is a stochastic disturbance.

In order to determine candidate control sequences, we ignore the stochastic disturbances and use feedback linearization to map between the original control inputs $u = (v_x, v_y)^T$ and new inputs $u_{lin} = (v, \omega)^T$. This map is given by

$$\begin{aligned} v &= v_x\cos(\theta) + v_y\sin(\theta), \\ \omega &= 1/\epsilon\,(-v_x\sin(\theta) + v_y\cos(\theta)), \end{aligned}$$

where ϵ is a tuning parameter related to the wheel base length.

We then check reachability for the simplified model $x_{k+1} = x_k + v_x$ and $y_{k+1} = y_k + v_y$, where v_x and v_y are the velocities in the x and y directions. This system is linear, so we can use TuLiP to check reachability [26]. We ignore θ here because our labels only depend on x and y.

The task for the robot is to sequentially visit two regions of interest while always remaining safe. Once the robot has visited the regions in order, it should return to the start and remain there. The atomic propositions are $\{home, unsafe, R1, R2\}$. The LTL specification formalizing this task is $\varphi = home \wedge \Diamond\Box home \wedge \Box\neg unsafe \wedge \Diamond(R1 \wedge \Diamond R2)$.

We used the procedure outlined in Section VI to create a finite MDP abstraction $\mathcal{M}$ of (15), where each state of $\mathcal{M}$ is a square region in $\mathcal{R}$. The actions at each state include transitioning to a neighbor, rotating in place, or not moving. As the system does not satisfy the Markov property (due to the θ term), we add four MDP states per region as a coarse discretization for θ. Due to symmetry of the regions and robot, we only need to calculate transitions for one region. An additional 16-fold speedup is possible by also exploiting rotational symmetries, but this is not implemented.

All computations were run on an 2.4 GHz dual core desktop desktop with 2 GB of RAM. It took 20 minutes to construct the MDP abstraction and uncertainty sets using 400 samples for the Monte Carlo simulation. We used large sample approximations to estimate β^a (14) for an uncertainty level of 0.9 [24]. The deterministic Rabin automaton representing φ has seven states. The product MDP has 597 states and 18455 transitions and took 1.1 seconds to compute. It took 25.4 seconds to compute $\mathcal{M}_{ssp}$. It took 44.3 seconds to generate an ϵ-suboptimal robust control policy with likelihood uncertainty sets. We also generated a non-robust control policy (no uncertainty sets), for comparison, 3.4 seconds. In both cases, $\epsilon = 1e-5$.

The calculated maximum probability of satisfying the specification was 0.763 for the non-robust policy and 0.526 for the robust policy. However, as the non-robust policy does not account for transition matrix uncertainties, it is likely to be overly optimistic when implemented on a real system. In this case, the non-robust policy only had a 0.491 probability of satisfying the specification given the worst-case transition matrices in the uncertainty set. Figure 1 shows sample trajectories of the robot using the robust and non-robust control policies.

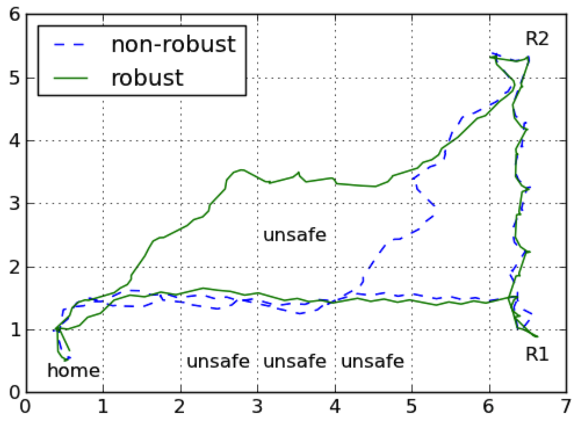

Fig. 1. Sample trajectories of the robot using the robust and non-robust control policies. The robust control policy takes a more conservative path.

VIII. Conclusions and Future Directions

We have presented a method for creating robust control policies for finite MDP abstractions of dynamical systems with temporal logic specifications. This robustness is useful both when the abstraction is not exact or when the dynamical system model does not adequately represent the physical system. Designing an ϵ-suboptimal robust control policy increases the time complexity only by a factor of $O(\log(1/\epsilon))$ compared to the non-robust policy for statistically meaningful uncertainty models.

In the future, we plan to extend these results to other temporal logics, such as probabilistic computational tree logic (PCTL). We are also looking into weakening the rectangularity assumption on the transition matrix uncertainty sets to allow for correlation. Finally, further work needs to be done on principled methods for abstraction of selected classes of dynamical systems as finite MDPs.

Acknowledgements

The authors would like to thank Scott Livingston for helpful comments. This work was supported by an NSF Graduate Research Fellowship and the Boeing Corporation.

Appendix

Theorem 2: The proof closely follows that of [5]. We add additional quantification over all possible probability distributions in (15) and (16). First, partition the state space. Let $S_1 = \{1\}$ and for $k = 2, 3, \dots,$ and define

$$S_k = \{i | i \notin S_1 \cup \dots \cup S_{k-1}\,,\ \min_{a \in A(i)} \ \max_{j \in S_1 \cup \dots \cup S_{k-1}} \ \min_{p \in \mathcal{P}_i^a} p_{ij} > 0\}. \tag{15}$$

Let m be the largest integer such that S_m is nonempty. It can be seen that the sets S_k cover the entire state space, i.e., $\cup_{k=1}^m S_k = S$.

Choose a vector $w > 0$ such that T is a contraction with respect to $\| \cdot \|_w$. Take the ith component w_i to be the same for states i in the same set S_k. We choose the components w_i of the vector w by

$$w_i = y_k \quad \text{if} \quad i \in S_k,$$

where $y_1, \dots, y_m$ are appropriately chosen scalars satisfying

$$1 = y_1 < y_2 < \dots < y_m.$$

Let

$$\epsilon := \min_{k=2,\dots,m} \ \min_{\mu \in M} \ \min_{i \in S_k} \ \min_{p \in \mathcal{P}_i^a} \sum_{j \in S_1 \cup \dots \cup S_{k-1}} P_{ij}^{\mu}, \tag{16}$$

and note that $0 < \epsilon \le 1$. We will show that it is sufficient to choose $y_2, \dots, y_m$ so that for some $\beta < 1$, we have

$$\frac{y_m}{y_k}(1-\epsilon) + \frac{y_{k-1}}{y_k}\epsilon \le \beta < 1, k = 2, \dots, m,$$

and then show that such a choice exists.

For all vectors $u, v \in \mathbb{R}^n$, select a μ such that $T_\mu u = Tu$. We then have for all i,

$$\begin{aligned}(Tv)(i) - (Tu)(i) &= (Tv)(i) - (T_\mu u)(i) \\ &\le (T_\mu v)(i) - (T_\mu u)(i) \\ &= \sum_{j=1}^{n} p_{ij}(\mu(i))(v(j) - u(j)).\end{aligned}$$

Let $k(j)$ be such that j belongs to the set $s_{k(j)}$. Then we have for any constant c,

$$\| v - u \|_w \le c \implies v(j) - u(j) \le c y_k, j = 2, \dots, n,$$

and thus for all i,

$$\begin{aligned}\frac{(Tv)(i)-(Tu)(i)}{cy_{k(i)}} &\leq \frac{1}{y_{k(i)}}\sum_{j=1}^{n} p_{ij}(\mu(i))y_{k(j)} \\ &\leq \frac{y_{k(i)-1}}{y_{k(i)}}\sum_{j\in S_1\cup\cdots\cup S_{k(i)-1}} p_{ij}(\mu(i)) \\ &+ \frac{y_m}{y_{k(i)}}\sum_{j\in S_{k(i)}\cup\cdots\cup S_m} p_{ij}(\mu(i)) \\ &= \left(\frac{y_{k(i)-1}}{y_{k(i)}} - \frac{y_m}{y_{k(i)}}\right)\sum_{j\in S_1\cup\cdots\cup S_{k(i)-1}} p_{ij}(\mu(i)) \\ &+ \frac{y_m}{y_{k(i)}} \leq \left(\frac{y_{k(i)-1}}{y_{k(i)}} - \frac{y_m}{y_{k(i)}}\right)\epsilon + \frac{y_m}{y_{k(i)}} \leq \beta.\end{aligned}$$

Thus, we have that

$$\frac{(Tv)(i)-(Tu)(i)}{w_i} \leq c\beta, \quad i = 1,\ldots,n,$$

which taking the max over i gives

$$\| Tv - Tu \|_w \leq c\beta,$$

for all $u, v \in \mathbb{R}^n$ with $\| u - v \|_w \leq c$.

Thus, we have that T is a contraction mapping under the $\| \cdot \|_w$ norm.

We now show that scalars $y_1, y_2, \ldots, y_m$ exist such that the above assumptions hold. Let $y_0 = 0, y_1 = 1$, and suppose that $y_1, y_2, \ldots, y_k$ have been chosen. If $\epsilon = 1$, we choose $y_{k+1} = y_k + 1$. If $\epsilon < 1$, we choose y_{k+1} to be

$$y_{k+1} = \frac{1}{2}(y_k + M_k)$$

where

$$M_k = \min_{1\leq i\leq k}\left\{y_i + \frac{\epsilon}{1-\epsilon}(y_i - y_{i-1})\right\}.$$

Using the fact that

$$M_k = \min\left\{M_k,\ y_{k+1} + \frac{\epsilon}{1-\epsilon}(y_{k+1} - y_k)\right\},$$

we have by induction that for all k, $y_k < y_{k+1} < M_{k+1}$ and thus we can construct the required sequence. ■

References

[1] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88(7):971–984, 2000.

[2] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Logic*, 5(1):1–25, 2004.

[3] J. A. Bagnell, A. Y. Ng, and J. G. Schneider. Solving uncertain Markov decision processes. Technical report, Carnegie Mellon University, 2001.

[4] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[5] D. P. Bertsekas. *Dynamic Programming and Optimal Control (Vol. I and II)*. Athena Scientific, 2001.

[6] O. Buffet. Reachability analysis for uncertain ssps. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, 2005.

[7] K. Chatterjee, K. Sen, and T. Henzinger. Model-checking omega-regular properties of interval Markov chains. In R. M. Amadio, editor, *Foundations of Software Science and Computation Structure (FoSSaCS)*, pages 302–317, March 2008.

[8] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the Association for Computing Machinery*, 42:857–907, 1995.

[9] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[10] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. LTL control in uncertain environments with probabilistic satisfaction guarantees. In *Proceedings of 18th IFAC World Congress*, 2011.

[11] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. MDP optimal control under temporal logic constraints. In *Proceedings of the IEEE Conference on Decision and Control*, 2011.

[12] E. A. Emerson. Temporal and modal logic. In *Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[13] R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122:71–109, 2000.

[14] B. Johnson and H. Kress-Gazit. Probabilistic analysis of correctness of high-level robot behavior with sensor error. In *Proceedings of Robotics: Science and Systems*, 2011.

[15] J. Klein and C. Baier. Experiments with deterministic omega-automata for formulas of linear temporal logic. *Theoretical Computer Science*, 363:182–195, 2006.

[16] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transaction on Automatic Control*, 53(1):287–297, 2008.

[17] H. Kress-Gazit, G. Fainekos, and G. Pappas. Where's Waldo? Sensor-based temporal logic motion planning. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3116–3121, 2007.

[18] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: verification of probabilistic real-time systems. In *Proceedings of 23rd International Conference on Computer Aided Verification*, 2011.

[19] M. Lahijanian, S. B. Andersson, and C. Belta. A probabilistic approach for control of a stochastic system from LTL specifications. In *IEEE Conference on Decision and Control*, 2009.

[20] M. Lahijanian, S. B. Andersson, and C. Belta. Control of Markov decision processes from PCTL specifications. In *Proceedings of the American Control Conference*, 2011.

[21] E. L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer, 2005.

[22] R. Majumdar, E. Render, and P. Tabuada. Robust discrete synthesis against unspecified disturbances. In *Proc. Hybrid Systems: Computation and Control*, pages 211–220, 2011.

[23] A. W. Naylor and G. R. Sell. *Linear Operator Theory in Engineering and Science*. Springer-Verlag, 1982.

[24] A. Nilim and L. El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53:780–798, 2005.

[25] J. K. Satia and R. E. L. Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):pp. 728–740, 1973.

[26] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon control for temporal logic specifications. In *Proc. of the 13th International Conference on Hybrid Systems: Computation and Control*, 2010.

[27] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 2010. submitted.

[28] D. Wu and X. Koutsoukos. Reachability analysis of uncertain systems using bounded-parameter Markov decision processes. *Artif. Intell.*, 172:945–954, 2008.