Control of Probabilistic Systems under Dynamic, Partially Known Environments with Temporal Logic Specifications

Tichakorn Wongpiromsarn and Emilio Frazzoli

Abstract— **We consider the synthesis of control policies for probabilistic systems, modeled by Markov decision processes, operating in partially known environments with temporal logic specifications. The environment is modeled by a set of Markov chains. Each Markov chain describes the behavior of the environment in each mode. The mode of the environment, however, is not known to the system. Two control objectives are considered: maximizing the expected probability and maximizing the worst-case probability that the system satisfies a given specification.**

I. Introduction

In many applications, control systems need to perform complex tasks and interact with their (potentially adversarial) environments. The correctness of these systems typically depends on the behaviors of the environments. For example, whether an autonomous vehicle exhibits a correct behavior at a pedestrian crossing depends on the behavior of the pedestrians, e.g., whether they actually cross the road, remain on the same side of the road or step in front of the vehicle while it is moving.

Temporal logics, which were primarily developed by the formal methods community for specifying and verifying correctness of software and hardware systems, have been recently employed to express complex behaviors of control systems. Its expressive power offers extensions to properties that can be expressed than safety and stability, typically studied in the control and hybrid systems domains. In particular, [1] shows that the traffic rule enforced in the 2007 DARPA Urban Challenge can be precisely described using these logics. Furthermore, the recent development of language equivalence and simulation notions allows abstraction of continuous systems to a purely discrete model [2]–[6]. This subsequently provides a framework for integrating methodologies from the formal methods and the control-theoretic communities and enables formal specification, design and verification of control systems with complex behaviors.

Controller synthesis from temporal logic specifications has been considered in [7]–[10], assuming static environments. Synthesis of reactive controllers that takes into account all the possible behaviors of dynamic environments can be found in [11], [12]. In this case, the environment is treated as an adversary and the synthesis problem can be viewed as a two-player game between the system and the environment: the environment attempts to falsify the specification while the system attempts to satisfy it [13]. In these works, the system is assumed to be deterministic, i.e., an available control action in each state enables exactly one transition. Controller synthesis for probabilistic systems such as Markov decision processes (MDP) has been considered in [14], [15]. However, these works assume that at any time instance, the state of the system, including the environment, as well as their (probabilistic) models are fully known. This may not be a valid assumption in many applications, especially those in which the discrete internal state of the environment is not directly measurable [16]. For example, in the pedestrian crossing problem previously described, the behavior of the pedestrians depends on their destination, which is typically not known to the system. Having to account for all the possible behaviors of the pedestrians with respect to all their possible destinations may lead to conservative results and in many cases, unrealizable specifications.

Partially observable Markov decision process (POMDP) provides a principled mathematical framework to cope with partial observability in stochastic domains [17]. Roughly, the main idea is to maintain a belief, which is defined as a probability distribution over all the possible states. POMDP algorithms then operate in the belief space. Unfortunately, it has been shown that solving POMDPs exactly is computationally intractable [18]. Hence, point-based algorithms have been developed to compute an approximate solution based on the computation over a representative set of points from the belief space rather than the entire belief space [19].

In this paper, we take an initial step towards solving POMDPs that are subject to temporal logic specifications. In particular, we consider the problem where a collection of possible environment models is available to the system. Different models correspond to different modes of the environment. However, the system does not know in which mode the environment is. In addition, the environment may change its mode during an execution subject to certain constraints. We consider two control objectives: maximizing the expected probability and maximizing the worst-case probability that the system satisfies a given temporal logic specification. The first objective is closely related to solving POMDPs as previously described whereas the second objective is closely related to solving uncertain MDPs [20]. However, for both problems, we aim at maximizing the expected or worst-case probability of satisfying a temporal logic specification, instead of maximizing the expected or worst-case reward as considered in the POMDP and MDP literature.

The main contribution of this paper is twofold. First, we show that the expectation-based synthesis problem can be formulated as a control policy synthesis problem for MDPs

T. Wongpiromsarn is with the Singapore-MIT Alliance for Research and Technology, Singapore 117543, Singapore. nok@smart.mit.edu

E. Frazzoli is with the Massachusetts Institute of Technology, Cambridge, MA 02139, USA. frazzoli@mit.edu

under temporal logic specifications and provide a complete solution to the problem. Second, we define a mathematical object called *adversarial Markov decision process* (AMDP) and show that the worst-case-based synthesis problem can be formulated as a control policy synthesis problem for AMDP. A complete solution to the control policy synthesis for AMDP is then provided. Finally, we show that the maximum worst-case probability that a given specification is satisfied does not depend on whether the controller and the adversary play alternatively or both the control and adversarial policies are computed at the beginning of an execution.

The rest of the paper is organized as follows: We provide useful definitions and descriptions of the formalisms in the following section. Section III is dedicated to the problem formulation. The expectation-based and the worst-case-based control policy synthesis are considered in Section IV and Section V, respectively. Section VI presents an example. Finally, Section VII concludes the paper and discusses future work.

II. Preliminaries

We consider systems that comprise stochastic components. In this section, we define the formalisms used to describe such systems and their desired properties. For a set X, we let $|X|$ denote the size of X and X^*, X^ω and X^+ denote the set of finite, infinite and nonempty finite strings of X.

A. Automata

Definition 1: A *deterministic Rabin automaton* (DRA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_{init}, Acc)$ where (a) Q is a finite set of states, (b) Σ is a finite set called alphabet, (c) $\delta : Q \times \Sigma \to Q$ is a transition function, (d) $q_{init} \in Q$ is the initial state, and (e) $Acc \subseteq 2^Q \times 2^Q$ is the acceptance condition. We use the relation notation, $q \xrightarrow{w} q'$ to denote $\delta(q, w) = q'$.

Consider an infinite string $\sigma = \sigma_0\sigma_1 \ldots \in \Sigma^\omega$. A *run* for σ in $\mathcal{A}$ is an infinite sequence of states $q_0q_1 \ldots q_n$ such that $q_0 = q_{init}$ and $q_i \xrightarrow{\sigma_i} q_{i+1}$ for all $i \geq 0$. A run is *accepting* if there exists a pair $(H, K) \in Acc$ such that (a) there exists $n \geq 0$ such that for all $m \geq n$, $q_m \notin H$, and (b) there exist infinitely many $n \geq 0$ such that $q_n \in K$.

A string $\sigma \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if there is an accepting run of σ in $\mathcal{A}$. The language *accepted* by $\mathcal{A}$, denoted by $\mathcal{L}_\omega(\mathcal{A})$, is the set of all accepted strings of $\mathcal{A}$.

B. Linear Temporal Logic

Linear temporal logic (LTL) is a powerful specification language for precisely expressing a wide range of properties of systems. An LTL formula is built up from a set Π of atomic propositions, the logic connectives $\neg$, $\vee$, $\wedge$ and $\Longrightarrow$ and the temporal modal operators $\bigcirc$ ("next"), $\Box$ ("always"), $\Diamond$ ("eventually") and $\mathcal{U}$ ("until"), and is interpreted on infinite strings $\sigma_0\sigma_1\sigma_2 \ldots$ where $\sigma_i \in 2^\Pi$ for all $i \geq 0$.

Given propositional formulas p_1 and p_2, examples of LTL formulas include $p_1 \,\mathcal{U}\, p_2$ (read as "p_1 until p_2"), which states that p_1 has to remain true until p_2 becomes true and there is some point in an execution where p_2 becomes true. A safety formula $\Box p_1$ (read as "always p_1") simply asserts that property p_1 remains invariantly true throughout an execution. A reachability formula $\Diamond p_1$ (read as "eventually p_1") states that property p_1 becomes true at least once in an execution (i.e., there exists a reachable state that satisfies p_1).

It can be shown that for any LTL formula φ over Π, there exists a DRA $\mathcal{A}$ with alphabet $\Sigma = 2^\Pi$ that accepts all and only words over Π that satisfy φ. Such $\mathcal{A}$ can be automatically constructed using existing tools [21]. We refer the reader to [22]–[24] for more details on LTL.

C. Systems and Control Policies

Definition 2: A *(discrete-time) Markov chain* (MC) is a tuple $\mathcal{M} = (S, \mathbf{P}, s_{init}, \Pi, L)$ where (a) S is a countable set of states, (b) $\mathbf{P} : S \times S \to [0, 1]$ is the transition probability function such that for any state $s \in S$, $\sum_{s' \in S} \mathbf{P}(s, s') = 1$, (c) $s_{init} \in S$ is the initial state, (d) Π is a set of atomic propositions, and (e) $L : S \to 2^\Pi$ is a labeling function.

Definition 3: A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$ where (a) S, s_{init}, Π and L are defined as in MC, (b) Act is a finite set of actions, and (c) $\mathbf{P} : S \times Act \times S \to [0, 1]$ is the transition probability function such that for any state $s \in S$ and action $\alpha \in Act$, $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\}$. An action α is *enabled* in state s if and only if $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$. We let $Act(s)$ denote the set of enabled actions in s.

Given a complete system as the composition of all its components, we are interested in computing a control policy for the system that optimizes certain objectives. We define a control policy for a system modeled by an MDP as follows.

Definition 4: Let $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$ be an MDP. A *control policy* for $\mathcal{M}$ is a function $\mathcal{C} : S^+ \to Act$ such that $\mathcal{C}(s_0s_1 \ldots s_n) \in Act(s_n)$ for all $s_0s_1 \ldots s_n \in S^+$.

Let $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$ be an MDP and $\mathcal{C} : S^+ \to Act$ be a control policy for $\mathcal{M}$. An infinite sequence $r^{\mathcal{C}}_{\mathcal{M}} = s_0s_1 \ldots$ on $\mathcal{M}$ generated under policy $\mathcal{C}$ is called a *path* on $\mathcal{M}$ if $s_0 = s_{init}$ and $\mathbf{P}(s_i, \mathcal{C}(s_0s_1 \ldots s_i), s_{i+1}) > 0$ for all i. The subsequence $s_0s_1 \ldots s_n$ where $n \geq 0$ is the *prefix* of length n of $r^{\mathcal{C}}_{\mathcal{M}}$. We define $Paths^{\mathcal{C}}_{\mathcal{M}}$ and $FPaths^{\mathcal{C}}_{\mathcal{M}}$ as the set of all infinite paths of $\mathcal{M}$ under policy $\mathcal{C}$ and their finite prefixes, respectively. For $s_0s_1 \ldots s_n \in FPaths^{\mathcal{C}}_{\mathcal{M}}$, we let $Paths^{\mathcal{C}}_{\mathcal{M}}(s_0s_1 \ldots s_n)$ denote the set of all paths in $Paths^{\mathcal{C}}_{\mathcal{M}}$ with prefix $s_0s_1 \ldots s_n$. It can be shown [22] that there exists a unique probability measure $Pr^{\mathcal{C}}_{\mathcal{M}}$ on the $\sigma-$algebra associated with $\mathcal{M}$ under policy $\mathcal{C}$ where for any $s_0s_1 \ldots s_n \in FPaths^{\mathcal{C}}_{\mathcal{M}}$,

$$\Pr^{\mathcal{C}}_{\mathcal{M}}\{Paths^{\mathcal{C}}_{\mathcal{M}}(s_0s_1 \ldots s_n)\} = \prod_{0 \leq i < n} \mathbf{P}(s_i, \mathcal{C}(s_0s_1 \ldots s_i), s_{i+1}). \quad (1)$$

In addition, the probability for $\mathcal{M}$ to satisfy an LTL formula φ under policy $\mathcal{C}$ can be defined as

$$\Pr^{\mathcal{C}}_{\mathcal{M}}(\varphi) = \Pr^{\mathcal{C}}_{\mathcal{M}}\{s_0s_1 \ldots \in Paths^{\mathcal{C}}_{\mathcal{M}} \mid L(s_0)L(s_1) \ldots \models \varphi\}.$$

For a given (possibly noninitial) state $s \in S$, we let $\mathcal{M}^s = (S, Act, \mathbf{P}, s, \Pi, L)$, i.e., $\mathcal{M}^s$ is the same as $\mathcal{M}$ except that its initial state is s. We define $\Pr^{\mathcal{C}}_{\mathcal{M}}(s \models \varphi) = \Pr^{\mathcal{C}}_{\mathcal{M}^s}(\varphi)$ as the probability for $\mathcal{M}$ to satisfy φ under policy $\mathcal{C}$, starting from s.

A control policy essentially resolves all the nondeterministic choices in an MDP and induces a Markov chain $\mathcal{M}_{\mathcal{C}}$ that formalizes the behavior of $\mathcal{M}$ under control policy $\mathcal{C}$ [22]. In

general, $\mathcal{M}_\mathcal{C}$ contains all the states in S^+ and hence may not be finite even though $\mathcal{M}$ is finite. However, for a special case where $\mathcal{C}$ is a memoryless or a finite memory control policy, it can be shown that $\mathcal{M}_\mathcal{C}$ can be identified with a finite MC. Roughly, a memoryless control policy always picks the action based only on the current state of $\mathcal{M}$, regardless of the path that led to that state. In contrast, a finite memory control policy also maintains its "mode" and picks the action based on its current mode and the current state of $\mathcal{M}$.

III. Problem Formulation

Consider a system that comprises 2 components: the plant and the environment. The system can regulate the state of the plant but has no control over the state of the environment. We assume that at any time instance, the state of the plant and the state of the environment, except its internal mode, can be precisely observed.

The plant is modeled by a finite MDP $\mathcal{M}^{pl} = (S^{pl}, Act, \mathbf{P}^{pl}, s^{pl}_{init}, \Pi^{pl}, L^{pl})$. We assume that for each $s \in S^{pl}$, there is an action $\alpha \in Act$ that is enabled in state s. In addition, we assume that the environment can be modeled by some MC in $\mathbf{M}^{env} = \{\mathcal{M}^{env}_1, \mathcal{M}^{env}_2, \ldots, \mathcal{M}^{env}_N\}$ where for each $i \in \{1, \ldots, N\}$, $\mathcal{M}^{env}_i = (S^{env}_i, \mathbf{P}^{env}_i, s^{env}_{init,i}, \Pi^{env}_i, L^{env}_i)$ is a finite MC that represents a possible model of the environment. For the simplicity of the presentation, we assume that for all $i \in \{1, \ldots, N\}$, $S^{env}_i = S^{env}$, $\Pi^{env}_i = \Pi^{env}$, $s^{env}_{init,i} = s^{env}_{init}$ and $L^{env}_i = L^{env}$; hence, $\mathcal{M}^{env}_1, \mathcal{M}^{env}_2, \ldots, \mathcal{M}^{env}_N$ differ only in the transition probability function. (If $S^{env}_i \neq S^{env}_j$ for some $i, j \in \{1, \ldots, N\}$, we define $S^{env} = \bigcup_i S^{env}_i \cup \{s_\phi\}$ where s_ϕ is a dummy state with $L^{env}(s_\phi) = \emptyset$ and only self transition. We can then expand each S^{env}_i to S^{env} by adding unreachable states, each with only transition to s_ϕ.) These different environment models can be considered as different *modes* of the environment. For the rest of the paper, we use "environment mode" to refer to some $\mathcal{M}^{env}_i \in \mathbf{M}^{env}$.

We further assume that the plant and the environment make a transition simultaneously, i.e., both of them make a transition at every time step. All $\mathcal{M}^{env}_i \in \mathbf{M}^{env}$ are available to the system. However, the system does not know exactly which $\mathcal{M}^{env}_i \in \mathbf{M}^{env}$ is the actual mode of the environment. Instead, it maintains the belief $\mathbf{B} : \mathbf{M}^{env} \to [0, 1]$, which is defined as a probability distribution over all possible environment modes such that $\sum_{1 \leq i \leq N} \mathbf{B}(\mathcal{M}^{env}_i) = 1$ and $\mathbf{B}(\mathcal{M}^{env}_i)$ returns the probability that $\mathcal{M}^{env}_i$ is the mode of the environment. The set of all the beliefs forms the belief space, denoted by $\mathbb{B}$. In order to obtain the belief at each time step, the system is given the initial belief $\mathbf{B}_{init} : \mathcal{M}^{env} \to [0, 1] \in \mathbb{B}$. Then, it subsequently updates the belief using a given belief update function $\tau : \mathbb{B} \times S^{env} \times S^{env} \to \mathbb{B}$ such that $\tau(\mathbf{B}, s, s')$ returns the belief after the environment makes a transition from state s with belief $\mathbf{B}$ to state s'. The belief update function can be defined based on the observation function as in the belief MDP construction for POMDPs [17].

In general, the belief space $\mathbb{B}$ may be infinite, rendering the control policy synthesis computationally intractable to solve exactly. To overcome this difficulty, we employ techniques for solving POMDPs and approximate $\mathbb{B}$ by a finite set of representative points from $\mathbb{B}$ and work with this approximate representation instead. Belief space approximation is beyond the scope of this paper and is subject to future work. Sampling techniques that have been proposed in the POMDP literature can be found in, e.g., [19], [25], [26].

Given a system model described by $\mathcal{M}^{pl}$, $\mathbf{M}^{env} = \{\mathcal{M}^{env}_1, \mathcal{M}^{env}_2, \ldots, \mathcal{M}^{env}_N\}$, the (finite) belief space $\mathbb{B}$, the initial belief $\mathbf{B}_{init}$, the belief update function τ and an LTL formula φ that describes the desired property of the system, we consider the following control policy synthesis problems.

Problem 1: Synthesize a control policy for the system that maximizes the expected probability that the system satisfies φ where the expected probability that the environment transitions from state $s \in S^{env}$ with belief $\mathbf{B} \in \mathbb{B}$ to state $s' \in S^{env}$ is given by $\sum_{1 \leq i \leq N} \mathbf{B}(\mathcal{M}^{env}_i) \mathbf{P}^{env}_i(s, s')$.

Problem 2: Synthesize a control policy for the system that maximizes the worst-case (among all the possible sequences of environment modes) probability that the system satisfies φ. The environment mode may change during an execution: when the environment is in state $s \in S^{env}$ with belief $\mathbf{B} \in \mathbb{B}$, it may switch to any mode $\mathcal{M}^{env}_i \in \mathbf{M}^{env}$ with $\mathbf{B}(\mathcal{M}^{env}_i) > 0$. We consider both the case where the controller and the environment plays a sequential game and the case where the control policy and the sequence of environment modes are computed before an execution.

Example 1: Consider a problem where an autonomous vehicle needs to navigate a road with a pedestrian walking on the pavement. The vehicle and the pedestrian are considered the plant and the environment, respectively. The pedestrian may or may not cross the road, depending on his/her destination, which is unknown to the system. Suppose the road is discretized into a finite number of cells $c_0, c_2, \ldots, c_M$. The vehicle is modeled by an MDP $\mathcal{M}^{pl} = (S^{pl}, Act, \mathbf{P}^{pl}, s^{pl}_{init}, \Pi^{pl}, L^{pl})$ whose state $s \in S^{pl}$ describes the cell occupied by the vehicle and whose action $\alpha \in Act$ corresponds to a motion primitive of the vehicle (e.g., cruise, accelerate, decelerate). The motion of the pedestrian is modeled by an MC $\mathcal{M}^{env}_i \in \mathbf{M}^{env}$ where $\mathbf{M}^{env} = \{\mathcal{M}^{env}_1, \mathcal{M}^{env}_2\}$. $\mathcal{M}^{env}_1 = (S^{env}, \mathbf{P}^{env}_1, s^{env}_{init}, \Pi^{env}, L^{env})$ represents the model of the pedestrian if s/he decides not to cross the road whereas $\mathcal{M}^{env}_2 = (S^{env}, \mathbf{P}^{env}_2, s^{env}_{init}, \Pi^{env}, L^{env})$ represents the model of the pedestrian if s/he decides to cross the road. A state $s \in S^{env}$ describes the cell occupied by the pedestrian. The labeling functions L^{pl} and L^{env} essentially map each cell to its label, with an index that identifies the vehicle from the pedestrian, i.e., $L^{pl}(c_j) = c^{pl}_j$ and $L^{env}(c_j) = c^{env}_j$ for all $j \in \{0, \ldots M\}$. Consider the desired property stating that the vehicle does not collide with the pedestrian until it reaches cell c_M (e.g., the end of the road). In this case, the specification φ can be written as $\varphi = \left(\neg \bigvee_{j \geq 0} (c^{pl}_j \wedge c^{env}_j)\right) \mathcal{U}\ c^{pl}_M$.

IV. Expectation-Based Control Policy Synthesis

To solve Problem 1, we first construct the MDP that represents the complete system, taking into account the uncertainties, captured by the belief, in the environment

mode. Then, we employ existing results in probabilistic verification and construct the product MDP and extract its optimal control policy. In this section, we describe these steps in more detail and discuss their connection to Problem 1.

A. Construction of the Complete System

Based on the notion of belief, we construct the complete environment model, represented by the MC $\mathcal{M}^{env} = (S^{env} \times \mathbb{B}, \mathbf{P}^{env}, \langle s^{env}_{init}, \mathbf{B}_{init}\rangle, \Pi^{env}, L^{env'})$ where for each $s, s' \in S^{env}$ and $\mathbf{B}, \mathbf{B}' \in \mathbb{B}$, $L^{env'}(s, \mathbf{B}) = L^{env}(s)$ and

$$\mathbf{P}^{env}(\langle s, \mathbf{B}\rangle, \langle s', \mathbf{B}'\rangle) = \begin{cases} \sum_i \mathbf{B}(\mathcal{M}^{env}_i)\mathbf{P}^{env}_i(s, s') & \\ \qquad \text{if } \tau(\mathbf{B}, s, s') = \mathbf{B}' & \\ 0 \qquad \text{otherwise} & \end{cases}, \tag{2}$$

It is straightforward to check that for all $\langle s, \mathbf{B}\rangle \in S^{env} \times \mathbb{B}$, $\sum_{s', \mathbf{B}'} \mathbf{P}^{env}(\langle s, \mathbf{B}\rangle, \langle s', \mathbf{B}'\rangle) = 1$, so $\mathcal{M}^{env}$ is a valid MC.

Assuming that the plant and the environment make a transition simultaneously, we obtain the complete system by constructing the synchronous parallel composition of the plant and the environment. Synchronous parallel composition of MDP and MC is defined as follows.

Definition 5: Let $\mathcal{M}_1 = (S_1, Act, \mathbf{P}_1, s_{init,1}, \Pi_1, L_1)$ be an MDP and $\mathcal{M}_2 = (S_2, \mathbf{P}_2, s_{init,2}, \Pi_2, L_2)$ be an MC. Their synchronous parallel composition, denoted by $\mathcal{M}_1 || \mathcal{M}_2$, is the MDP $\mathcal{M} = (S_1 \times S_2, Act, \mathbf{P}, \langle s_{init,1}, s_{init,2}\rangle, \Pi_1 \cup \Pi_2, L)$ where:

- For each $s_1, s'_1 \in S_1$, $s_2, s'_2 \in S_2$ and $\alpha \in Act$, $\mathbf{P}(\langle s_1, s_2\rangle, \alpha, \langle s'_1, s'_2\rangle) = \mathbf{P}_1(s_1, \alpha, s'_1)\mathbf{P}_2(s_2, s'_2)$.
- For each $s_1 \in S_1$ and $s_2 \in S_2$, $L(\langle s_1, s_2\rangle) = L_1(s_1) \cup L_2(s_2)$.

Hence, our complete system can be modeled by the MDP $\mathcal{M}^{pl} || \mathcal{M}^{env}$. We denote this MDP by $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$. Note that a state $s \in S$ is of the form $s = \langle s^{pl}, s^{env}, \mathbf{B}\rangle$ where $s^{pl} \in S^{pl}$, $s^{env} \in S^{env}$ and $\mathbf{B} \in \mathbb{B}$. The following lemma shows that Problem 1 can be solved by finding a control policy $\mathcal{C}$ for $\mathcal{M}$ that maximizes $\Pr^{\mathcal{C}}_{\mathcal{M}}(\varphi)$.

Lemma 1: Let $r^{\mathcal{C}}_{\mathcal{M}} = s_0 s_1 \dots s_n$ be a finite path of $\mathcal{M}$ under policy $\mathcal{C}$ where for each i, $s_i = \langle s^{pl}_i, s^{env}_i, \mathbf{B}_i\rangle \in S^{pl} \times S^{env} \times \mathbb{B}$. Then,

$$\Pr^{\mathcal{C}}_{\mathcal{M}}\{Path^{\mathcal{C}}_{\mathcal{M}}(r^{\mathcal{C}}_{\mathcal{M}})\} = \prod_{0 \le j < n} \left(\mathbf{P}(s^{pl}_j, \mathcal{C}(s_0 s_1 \dots s_j), s^{pl}_{j+1}) \sum_{1 \le i \le N} \mathbf{B}_j(\mathcal{M}^{env}_i)\mathbf{P}^{env}_i(s^{env}_j, s^{env}_{j+1}) \right) \tag{3}$$

Hence, given an LTL formula φ, $\Pr^{\mathcal{C}}_{\mathcal{M}}(\varphi)$ gives the expected probability that the system satisfies φ under policy $\mathcal{C}$.

Proof: The proof straightforwardly follows from the definition of $\mathcal{M}^{env}$ and $\mathcal{M}$. ∎

B. Construction of the Product MDP

Let $\mathcal{A}_\varphi = (Q, 2^\Pi, \delta, q_{init}, Acc)$ be a DRA that recognizes the specification φ. Our next step is to obtain a finite MDP $\mathcal{M}_p = (S_p, Act_p, \mathbf{P}_p, s_{p,init}, \Pi_p, L_p)$ as the product of $\mathcal{M}$ and $\mathcal{A}_\varphi$, defined as follows.

Definition 6: Let $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$ be an MDP and let $\mathcal{A} = (Q, 2^\Pi, \delta, q_{init}, Acc)$ be a DRA. Then, the product of $\mathcal{M}$ and $\mathcal{A}$ is the MDP $\mathcal{M}_p = \mathcal{M} \otimes \mathcal{A}$ defined by $\mathcal{M}_p = (S_p, Act, \mathbf{P}_p, s_{p,init}, \Pi_p, L_p)$ where $S_p = S \times Q$, $s_{p,init} = \langle s_{init}, \delta(q_{init}, L(s_{init}))\rangle$, $\Pi_p = Q$, $L_p(\langle s, q\rangle) = \{q\}$ and

$$\mathbf{P}_p(\langle s, q\rangle, \alpha, \langle s', q'\rangle) = \begin{cases} \mathbf{P}(s, \alpha, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Consider a path $r^{\mathcal{C}_p}_{\mathcal{M}_p} = \langle s_0, q_0\rangle\langle s_1, q_1\rangle \dots$ of $\mathcal{M}_p$ under some control policy $\mathcal{C}_p$. We say that $r^{\mathcal{C}_p}_{\mathcal{M}_p}$ is accepting if and only if there exists a pair $(H, K) \in Acc$ such that the word generated by $r^{\mathcal{C}_p}_{\mathcal{M}_p}$ intersects with H finitely many times and intersects with K infinitely many times, i.e., (1) there exists $n \ge 0$ such that for all $m \ge n$, $L_p(\langle s_m, q_m\rangle) \cap H = \emptyset$, and (2) there exists infinitely many $n \ge 0$ such that $L_p(\langle s_n, q_n\rangle) \cap K \ne \emptyset$.

Stepping through the above definition shows that given a path $r^{\mathcal{C}_p}_{\mathcal{M}_p} = \langle s_0, q_0\rangle\langle s_1, q_1\rangle \dots$ of $\mathcal{M}_p$ generated under some control policy $\mathcal{C}_p$, the corresponding path $s_0 s_1 \dots$ on $\mathcal{M}$ generates a word $L(s_0)L(s_1)\dots$ that satisfies φ if and only if $r^{\mathcal{C}_p}_{\mathcal{M}_p}$ is accepting. Therefore, each accepting path of $\mathcal{M}_p$ uniquely corresponds to a path of $\mathcal{M}$ whose word satisfies φ. In addition, a control policy $\mathcal{C}_p$ on $\mathcal{M}_p$ induces a corresponding control policy $\mathcal{C}$ on $\mathcal{M}$. The details for generating $\mathcal{C}$ from $\mathcal{C}_p$ can be found, e.g., in [14], [22].

C. Control Policy Synthesis for Product MDP

From probabilistic verification, it has been shown that the maximum probability for $\mathcal{M}$ to satisfy φ is equivalent to the maximum probability of reaching a certain set of states of $\mathcal{M}_p$ known as *accepting maximal end components* (AMECs). An *end component* of the product MDP $\mathcal{M}_p = (S_p, Act_p, \mathbf{P}_p, s_{p,init}, \Pi_p, L_p)$ is a pair (T, A) where $\emptyset \ne T \subseteq S_p$ and $A : T \to 2^{Act_p}$ such that (1) $\emptyset \ne A(s) \subseteq Act_p(s)$ for all $s \in T$, (2) the directed graph induced by (T, A) is strongly connected, and (3) for all $s \in T$ and $\alpha \in A(s)$, $\{t \in S_p \mid \mathbf{P}_p(s, \alpha, t) > 0\} \subseteq T$. An *accepting maximal end component* of $\mathcal{M}_p$ is an end component (T, A) such that for some $(H, K) \in Acc$, $H \cap T = \emptyset$ and $K \cap T \ne \emptyset$ and there is no end component $(T', A') \ne (T, A)$ such that $T \subseteq T'$ and $A(s) \subseteq A'(s)$ for all $s \in T$. It has an important property that starting from any state in T, there exists a finite memory control policy to keep the state within T forever while visiting all states in T infinitely often with probability 1. AMECs of $\mathcal{M}_p$ can be efficiently identified based on iterative computations of strongly connected components of $\mathcal{M}_p$. We refer the reader to [22] for more details.

Once the AMECs of $\mathcal{M}_p$ are identified, we then compute the maximum probability of reaching S_G where S_G contains all the states in the AMECs of $\mathcal{M}_p$. For the rest of the paper, we use an LTL-like notations to describe events in MDPs. In particular, we use $\Diamond S_G$ to denote the event of reaching some state in S_G eventually.

For each $s \in S_p$, let x_s denote the maximum probability of reaching a state in S_G, starting from s. Formall, $x_s = \sup_{\mathcal{C}_p} \Pr^{\mathcal{C}_p}_{\mathcal{M}_p}(s \models \Diamond S_G)$. There are two main techniques for computing the probability x_s for each $s \in S_p$: linear programming (LP) and value iteration. LP-based techniques yield an exact solution but they typically do not scale as well as value iteration. On the other hand, value iteration

is an iterative numerical technique. This method works by successively computing the probability vector $(x_s^{(k)})_{s\in S_p}$ for increasing $k \geq 0$ such that $\lim_{k\to\infty} x_s^{(k)} = x_s$ for all $s \in S_p$. Initially, we set $x_s^{(0)} = 1$ if $s \in S_G$ and $x_s^{(0)} = 0$ otherwise. In the $(k+1)$th iteration where $k \geq 0$, we set

$$x_s^{(k+1)} = \begin{cases} 1 & \text{if } s \in S_G \\ \max\limits_{\alpha\in Act_p(s)} \sum\limits_{t\in S_p} \mathbf{P}_p(s,\alpha,t)x_t^{(k)} & \text{otherwise.} \end{cases} \quad (5)$$

In practice, we terminate the computation and say that $x_s^{(k)}$ converges when a termination criterion such as $\max_{s\in S_p} |x_s^{(k+1)} - x_s^{(k)}| < \epsilon$ is satisfied for some fixed (typically very small) threshold ϵ.

Once the vector $(x_s)_{s\in S_p}$ is computed, a finite memory control policy $\mathcal{C}_p$ for $\mathcal{M}_p$ that maximizes the probability for $\mathcal{M}$ to satisfy φ can be constructed as follows. First, consider the case when $\mathcal{M}_p$ is in state $s \in S_G$. In this case, s belongs to some AMEC (T, A) and the policy $\mathcal{C}_p$ selects an action $\alpha \in A(s)$ such that all actions in $A(s)$ are scheduled infinitely often. (For example, $\mathcal{C}_p$ may select the action for s according to a round-robin policy.) Next, consider the case when $\mathcal{M}_p$ is in state $s \in S_p \setminus S_G$. In this case, $\mathcal{C}_p$ picks an action to ensure that $\Pr_{\mathcal{M}}^{\mathcal{C}_p}(s \models \Diamond S_G) = x_s$ can be achieved. If $x_s = 0$, an action in $Act_p(s)$ can be chosen arbitrarily. Otherwise, $\mathcal{C}_p$ picks an action $\alpha \in Act_p^{max}(s)$ such that $\mathbf{P}_p(s,\alpha,t) > 0$ for some $t \in S_p$ with $\|t\| = \|s\| - 1$. Here, $Act_p^{max}(s) \subseteq Act_p(s)$ is the set of actions such that for all $\alpha \in Act_p^{max}(s)$, $x_s = \sum_{t\in S_p} \mathbf{P}(s,\alpha,t)x_t$ and $\|s\|$ denotes the length of a shortest path from s to a state in S_G, using only actions in Act_p^{max}.

It can be shown [22] that the complexity of the control policy synthesis procedure outlined above is polynomial in $|S_p| = |S^{pl}||S^{env}||\mathbb{B}||Q|$.

V. Worst-Case-Based Control Policy Synthesis

To solve Problem 2, we first propose a mathematical object called *adversarial Markov decision process* (AMDP). Then, we show that Problem 2 can be formulated as finding an optimal control policy for an AMDP. Finally, control policy synthesis for AMDP is discussed.

A. Adversarial Markov Decision Process

Definition 7: An *adversarial Markov decision process* (AMDP) is a tuple $\mathcal{M}^{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}, s_{init}, \Pi, L)$ where S, s_{init}, Π and L are defined as in MDP and

- Act_C is a finite set of control actions,
- Act_A is a finite set of adversarial actions, and
- $\mathbf{P} : S \times Act_C \times Act_A \times S \to [0,1]$ is the transition probability function such that for any $s \in S$, $\alpha \in Act_C$ and $\beta \in Act_A$, $\sum_{t\in S} \mathbf{P}(s,\alpha,\beta,t) \in \{0,1\}$.

We say that a control action α is *enabled* in state s if and only if there exists an adversarial action β such that $\sum_{t\in S} \mathbf{P}(s,\alpha,\beta,t) = 1$. Similarly, an adversarial action β is *enabled* in state s if and only if there exists a control action α such that $\sum_{t\in S} \mathbf{P}(s,\alpha,\beta,t) = 1$. Let $Act_C(s)$ and $Act_A(s)$ denote the set of enabled control and adversarial actions in s. We assume that for all $s \in S$, $\alpha \in Act_C(s)$ and $\beta \in Act_A(s)$, $\sum_{t\in S} \mathbf{P}(s,\alpha,\beta,t) = 1$, i.e., whether an adversarial (resp. control) action is enabled in state s depends only on the state s itself but not on a control (resp. adversarial) action taken by the system (resp. adversary).

Given an AMDP $\mathcal{M}^{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}, s_{init}, \Pi, L)$, a control policy $\mathcal{C} : S^+ \to Act_C$ and an adversarial policy $\mathcal{D} : S^+ \to Act_A$ for an AMDP can be defined such that $\mathcal{C}(s_0 s_1 \dots s_n) \in Act_C(s_n)$ and $\mathcal{D}(s_0 s_1 \dots s_n) \in Act_A(s_n)$ for all $s_0 s_1 \dots s_n \in S^+$. A unique policy measure $\Pr_{\mathcal{M}^{\mathcal{A}}}^{\mathcal{C},\mathcal{D}}$ on the σ-algebra associated with $\mathcal{M}^{\mathcal{A}}$ under control policy $\mathcal{C}$ and adversarial policy $\mathcal{D}$ can then be defined based on the notion of path on $\mathcal{M}^{\mathcal{A}}$ as for an ordinary MDP.

We end the section with important properties of AMDP that will be employed in the control policy synthesis. Due to space constraints, we only provide a sketch of proof here. The detailed proof can be found in a technical report [27].

Proposition 1: Let $\mathcal{M}^{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}, s_{init}, \Pi, L)$ be a finite AMDP and $S_G \subseteq S$ be the set of goal states. Let

$$x_s = \sup_{\mathcal{C}_0\in\mathbb{C}_0} \inf_{\mathcal{D}_0\in\mathbb{D}_0} \sup_{\mathcal{C}_1\in\mathbb{C}_1} \inf_{\mathcal{D}_1\in\mathbb{D}_1} \dots \Pr_{\mathcal{M}^{\mathcal{A}}}^{\mathcal{C},\mathcal{D}}(s \models \Diamond S_G), \quad (6)$$

where for any $n \geq 0$, $\mathbb{C}_n = \{\mathcal{C} : S^{n+1} \to Act_C \mid \mathcal{C}(s_0 s_1 \dots s_n) \in Act_C(s_n)\}$, $\mathbb{D}_n = \{\mathcal{D} : S^{n+1} \to Act_A \mid \mathcal{D}(s_0 s_1 \dots s_n) \in Act_A(s_n)\}$, $\mathcal{C}(s_0 s_1 \dots s_n) = \mathcal{C}_n(s_0 s_1 \dots s_n)$ and $\mathcal{D}(s_0 s_1 \dots s_n) = \mathcal{D}_n(s_0 s_1 \dots s_n)$. For each $k \geq 0$, consider a vector $(x_s^{(k)})_{s\in S}$ where $x_s^{(0)} = 1$ for all $s \in S_G$, $x_s^{(0)} = 0$ for all $s \notin S_G$ and for all $k \geq 0$,

$$x_s^{(k+1)} = \begin{cases} 1 & \text{if } s \in S_G \\ \max\limits_{\alpha\in Act_C(s)} \min\limits_{\beta\in Act_A(s)} \sum\limits_{t\in S} \mathbf{P}(s,\alpha,\beta,t)x_t^{(k)} & \text{otherwise} \end{cases} \quad (7)$$

Then, for any $s \in S$, $x_s^{(0)} \leq x_s^{(1)} \leq \dots \leq x_s$ and $x_s = \lim_{k\to\infty} x_s^{(k)}$.

Proof: Roughly, we first prove that in this case, perfect duality holds. Then, we show, using induction on k, that for any $k \geq 0$ and $s \in S$,

$$\begin{aligned} x_s^{(k)} = {} & \sup_{\mathcal{C}_0\in\mathbb{C}_0} \inf_{\mathcal{D}_0\in\mathbb{D}_0} \sup_{\mathcal{C}_1\in\mathbb{C}_1} \inf_{\mathcal{D}_1\in\mathbb{D}_1} \dots \\ & \sup_{\mathcal{C}_{k-1}\in\mathbb{C}_{k-1}} \inf_{\mathcal{D}_{k-1}\in\mathbb{D}_{k-1}} \Pr_{\mathcal{M}^{\mathcal{A}}}^{\mathcal{C},\mathcal{D}}(s \models \Diamond^{\leq k} S_G), \end{aligned}$$

where $\mathcal{C}$ and $\mathcal{D}$ are control and adversarial policies such that for any n such that $0 \leq n < k$, $\mathcal{C}(s_0 s_1 \dots s_n) = \mathcal{C}_n(s_0 s_1 \dots s_n)$ and $\mathcal{D}(s_0 s_1 \dots s_n) = \mathcal{D}_n(s_0 s_1 \dots s_n)$. Since the set of events $\Diamond^{\leq k+1} S_G$ includes the set of events $\Diamond^{\leq k} S_G$, we obtain $x_s^{(k)} \leq x_s^{(k+1)}$. Finally, we can conclude the proof using the fact that the sequence $x_s^{(0)}, x_s^{(1)}, \dots$ is monotonic and bounded (and hence has a finite limit) and $\Diamond S_G$ is the countable union of the events $\Diamond^{\leq k} S_G$. ∎

Proposition 2: Let $\mathcal{M}^{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}, s_{init}, \Pi, L)$ be a finite AMDP and $S_G \subseteq S$ be the set of goal states. Let

$$y_s = \sup_{\mathcal{C}} \inf_{\mathcal{D}} \Pr_{\mathcal{M}^{\mathcal{A}}}^{\mathcal{C},\mathcal{D}}(s \models \Diamond S_G). \quad (8)$$

For each $k \geq 0$, consider a vector $(y_s^{(k)})_{s\in S}$ where $y_s^{(0)} = 1$ for all $s \in S_G$, $y_s^{(0)} = 0$ for all $s \notin S_G$ and for all $k \geq 0$,

$$y_s^{(k+1)} = \begin{cases} 1 & \text{if } s \in S_G \\ \max\limits_{\alpha\in Act_C(s)} \min\limits_{\beta\in Act_A(s)} \sum\limits_{t\in S} \mathbf{P}(s,\alpha,\beta,t)y_t^{(k)} & \text{otherwise} \end{cases} \quad (9)$$

Then, for any $s \in S$, $y_s^{(0)} \leq y_s^{(1)} \leq \ldots \leq y_s$ and $y_s = \lim_{k\to\infty} y_s^{(k)}$.

Proof: The proof closely follows the proof of Proposition 1. Details are available in a technical report [27]. ■

From Proposition 1 and Proposition 2, we can conclude that the sequential game (6) is equivalent to its nonsequential counterpart (8).

Corollary 1: The maximum worst-case probability of reaching a set S_G of states in an AMDP $\mathcal{M}^{\mathcal{A}}$ does not depend on whether the controller and the adversary play alternatively or both the control and adversarial policies are computed at the beginning of an execution.

B. The Complete System as an AMDP

We start by constructing an MDP $\mathcal{M} = (S, Act, \mathbf{P}, s_{init}, \Pi, L)$ that represents the complete system as described in Section IV-A. As discussed earlier, a state of $\mathcal{M}$ is of the form $\langle s^{pl}, s^{env}, \mathbf{B}\rangle$ where $s^{pl} \in S^{pl}$, $s^{env} \in S^{env}$ and $\mathbf{B} \in \mathbb{B}$. The corresponding AMDP $\mathcal{M}_{\mathcal{A}}$ of $\mathcal{M}$ is then defined as $\mathcal{M}_{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}_{\mathcal{A}}, s_{init}, \Pi, L)$ where $Act_C = Act$, $Act_A = \{\beta_1, \ldots, \beta_N\}$ (i.e., β_i corresponds to the environment choosing mode $\mathcal{M}_i^{env}$) and for any $s_1^{pl}, s_2^{pl} \in S^{pl}$, $s_1^{env}, s_2^{env} \in S^{env}$, $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{B}$, $\alpha \in Act_C$ and $1 \leq i \leq N$,

$$\mathbf{P}_{\mathcal{A}}(\langle s_1^{pl}, s_1^{env}, \mathbf{B}_1\rangle, \alpha, \beta_i, \langle s_2^{pl}, s_2^{env}, \mathbf{B}_2\rangle) = \begin{cases} \mathbf{P}^{pl}(s_1^{pl}, \alpha, s_2^{pl})\mathbf{P}_i^{env}(s_1^{env}, s_2^{env}) & \text{if } \mathbf{B}_1(\mathcal{M}_i^{env}) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad (10)$$

if $\tau(\mathbf{B}_1, s, s') = \mathbf{B}_2$; otherwise, $\mathbf{P}_{\mathcal{A}}(\langle s_1^{pl}, s_1^{env}, \mathbf{B}_1\rangle, \alpha, \beta_i, \langle s_2^{pl}, s_2^{env}, \mathbf{B}_2\rangle) = 0$.

It is straightforward to check that $\mathcal{M}_{\mathcal{A}}$ is a valid AMDP. Furthermore, based on this construction and the assumptions that (1) at any plant state $s^{pl} \in S^{pl}$, there exists an action that is enabled in s^{pl}, and (2) at any point in an execution, the belief $\mathbf{B} \in \mathbb{B}$ satisfies $\sum_{1\leq i\leq N} \mathbf{B}(\mathcal{M}_i^{env}) = 1$, it can be shown that at any state $s \in S$, there exists a control action $\alpha \in Act_C$ and an adversarial action $\beta \in Act_A$ that are enabled in s. In addition, consider the case where the environment is in state $s^{env} \in S^{env}$ with belief $\mathbf{B} \in \mathbb{B}$. It can be shown that for all $\mathcal{M}_i^{env} \in \mathbf{M}^{env}$, if $\mathbf{B}(\mathcal{M}_i^{env}) > 0$, then β_i is enabled in $\langle s^{pl}, s^{env}, \mathbf{B}\rangle$ for all $s^{pl} \in S^{pl}$. Thus, we can conclude that $\mathcal{M}_{\mathcal{A}}$ represents the complete system for Problem 2.

Remark 1: According to (10), the system does not need to maintain the exact belief in each state. The only information needed to construct an AMDP that represents the complete system is all the possible modes of the environment in each state of the complete system. This allows us to integrate methodologies for discrete state estimation [16] to reduce the size of the AMDP. This is subject to future work.

C. Control Policy Synthesis for AMDP

Similar to control policy synthesis for MDP, control policy synthesis for AMDP $\mathcal{M}^{\mathcal{A}}$ can be done on the basis of a product construction. The product of $\mathcal{M}^{\mathcal{A}} = (S, Act_C, Act_A, \mathbf{P}, s_{init}, \Pi, L)$ and DRA $\mathcal{A} = (Q, 2^{\Pi}, \delta, q_{init}, Acc)$ is an AMDP $\mathcal{M}_p^{\mathcal{A}} = (S_p, Act_C, Act_A, \mathbf{P}_p, s_{p,init}, \Pi_p, L_p)$, which is defined similar to the product of MDP and DRA, except that the set of actions is partitioned into the set of control and the set of adversarial actions.

Following the steps for synthesizing a control policy for product MDP, we identify the AMECs of $\mathcal{M}_p^{\mathcal{A}}$. An AMEC of $\mathcal{M}_p^{\mathcal{A}}$ is defined based on the notion of end component as for the case of product MDP. However, an end component of $\mathcal{M}_p^{\mathcal{A}}$ needs to be defined, taking into account the adversary. Specifically, an end component of $\mathcal{M}_p^{\mathcal{A}}$ is a pair (T, A) where $\emptyset \neq T \subseteq S_p$ and $A : T \to 2^{Act_C}$ such that (1) $\emptyset \neq A(s) \subseteq Act_C(s)$ for all $s \in T$, (2) the directed graph induced by (T, A) under any adversarial policy is strongly connected, and (3) for all $s \in T$, $\alpha \in A(s)$ and $\beta \in Act_A(s)$, $\{t \in S_p \mid \mathbf{P}_p(s, \alpha, \beta, t) > 0\} \subseteq T$.

Using a similar argument as in the case of product MDP [22], it can be shown that the maximum worst-case probability for $\mathcal{M}^{\mathcal{A}}$ to satisfy φ is equivalent to the maximum worst-case probability of reaching a states in AMECs of $\mathcal{M}_p^{\mathcal{A}}$. We can then apply Proposition 1 and 2 to compute x_s, which is equivalent to y_s, using value iteration. A control policy for $\mathcal{M}_p^{\mathcal{A}}$ that maximizes the worst-case probability for $\mathcal{M}^{\mathcal{A}}$ to satisfy φ can be constructed as outlined at the end of Section IV-C for product MDP. Similar to the expectation-based synthesis, the complexity of this control policy synthesis procedure is polynomial in the size of $\mathcal{M}_p^{\mathcal{A}}$.

VI. Example

Consider, once again, the autonomous vehicle problem described in Example 1. Suppose the road is discretized into 9 cells $c_0, \ldots, c_8$ as shown in Figure 1. The vehicle starts in cell c_0 and has to reach cell c_8 whereas the pedestrian starts in cell c_1. The models of the vehicle and the pedestrian are shown in Figure 2. The vehicle has two actions α_1 and α_2, which correspond to decelerating and accelerating, respectively. The pedestrian has 2 modes $\mathcal{M}_1^{env}$ and $\mathcal{M}_2^{env}$, which correspond to the cases where s/he wants to remain on the left side of the road and cross the road, respectively. A DRA $\mathcal{A}_{\varphi}$ that accepts all and only words that satisfy $\varphi = \left(\neg \bigvee_{j\geq 0}(c_j^{pl} \wedge c_j^{env})\right) \mathcal{U}\ c_8^{pl}$ is shown in Figure 3. Finally, we consider the set $\mathbb{B} = \{\mathbf{B}_0, \ldots, \mathbf{B}_8\}$ of beliefs where for all i, $\mathbf{B}_i(\mathcal{M}_1^{env}) = 0.1i$ and $\mathbf{B}_i(\mathcal{M}_2^{env}) = 1 - 0.1i$. We set $\mathbf{B}_{init} = \mathbf{B}_6$ where it is equally likely that the pedestrian is in mode $\mathcal{M}_1^{env}$ or mode $\mathcal{M}_2^{env}$. The belief update function τ is defined such that the longer the pedestrian stay on the left side of the road, the probability that s/he is in mode $\mathcal{M}_1^{env}$ increases. Once the pedestrian starts crossing the road, we change the belief to $\mathbf{B}_0$ where it is certain that the pedestrian is in mode $\mathcal{M}_2^{env}$. Specifically, for all i, we let

$$\tau(\mathbf{B}_i, s, s') = \begin{cases} \mathbf{B}_0 & \text{if } s' \in \{c_2, c_4, c_6, c_7\} \\ \mathbf{B}_i & \text{if } i = 0 \text{ or } i = 8 \\ \mathbf{B}_{i+1} & \text{otherwise} \end{cases}. \quad (11)$$

Both the expectation-based and the worst-case-base control policy synthesis procedures as described in Section IV and V are implemented in MATLAB. The computation was performed on a MacBook Pro with a 2.8 GHz Intel Core 2 Duo processor.

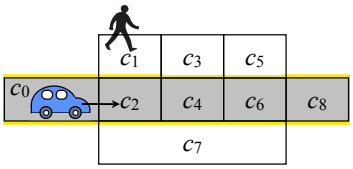
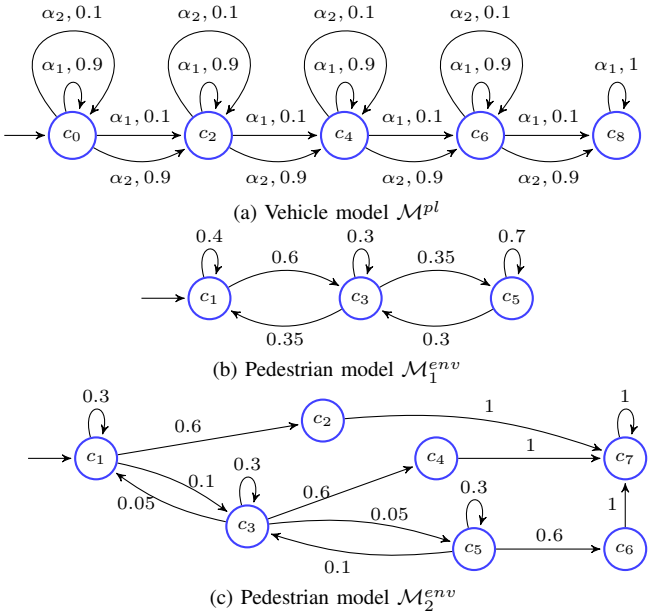


Fig. 1. The road and its partition used in the autonomous vehicle example.



(a) Vehicle model $\mathcal{M}^{pl}$

(b) Pedestrian model $\mathcal{M}_1^{env}$

(c) Pedestrian model $\mathcal{M}_2^{env}$

Fig. 2. Vehicle and pedestrian models.

First, we consider the expectation-based control policy synthesis (i.e., Problem 1). As outlined in Section IV, we first construct the MDP that represents the complete system. After removing all the unreachable states, the resulting MDP contains 65 states and the product MDP contains 53 states. The computation time is summarized in Table I. Note that the computation, especially the product MDP construction, can be sped up significantly if a more efficient representation of DRA is used. The maximum expected probability for the system to satisfy φ is 0.9454. Examination of the resulting control policy shows that this maximum expected probability of satisfying φ can be achieved by applying action α_1, i.e., decelerating, until the pedestrian crosses the street or the vehicle is not behind the pedestrian in the longitudinal direction, i.e., when the vehicle is in cell c_i and the pedestrian is in cell c_j where $j < i$. (Based on the expectation, the probability that the pedestrian eventually crosses the road is 1 according to the probability measure defined in (1).) If we include the belief $\mathbf{B}$ where $\mathbf{B}(\mathcal{M}_1^{env}) = 1$ and $\mathbf{B}(\mathcal{M}_2^{env}) = 0$, then the expectation-based optimal control policy is such that the vehicle applies α_1 until either the pedestrian crosses the road, the vehicle is not behind the pedestrian in the longitudinal direction or the belief is updated to $\mathbf{B}$, at which point, it applies α_2. Once the vehicle reaches the destination c_8, it applies α_1 forever.

Next, we consider the worst-case-based control policy synthesis (i.e., Problem 2). In this case, the resulting AMDP contains 49 states and the product AMDP contains 53 states after removing all the unreachable states. The maximum

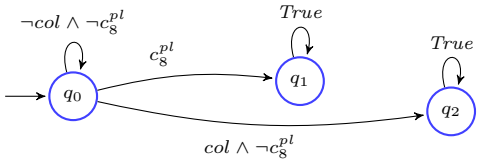


Fig. 3. A DRA $\mathcal{A}_\varphi$ that recognizes the prefixes of $\varphi = \neg col \, \mathcal{U} \, c_8^{pl}$ where the collision event col is defined as $col = \bigvee_{j \geq 0}(c_j^{pl} \wedge c_j^{env})$. The acceptance condition is $Acc = \{(\emptyset, \{q_1\})\}$.

	MDP / AMDP	product MDP / AMDP	Prob vector	Control policy	Total
Expectation	0.05	2.31	0.73	0.08	3.17
Worst-case	0.20	2.73	0.46	0.05	3.44

TABLE I

TIME REQUIRED (IN SECONDS) FOR EACH STEP OF COMPUTATION.

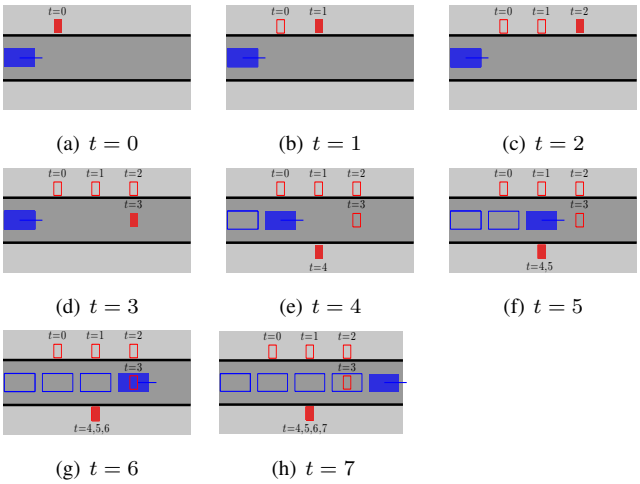
worst-case probability for the system to satisfy φ is 0.9033. The resulting control policy is slightly more aggressive than the expectation-based policy. In addition to the cases where the expectation-based controller applies α_2, the worst-case-based controller also applies α_2 when the vehicle is in c_0 and the pedestrian is in c_3 and when the vehicle is in c_2 and the pedestrian is in c_5.

Simulation results are shown in Figure 4 and Figure 5. The smaller (red) rectangle represents the pedestrian whereas the bigger (blue) rectangle represents the vehicle. The filled and unfilled rectangles represent their current positions and the trace of their trajectories, respectively. Notice that the vehicle successfully reaches its goal without colliding with the pedestrian, as required by its specification, with the worst-case-based controller being slightly more aggressive.

Finally, we would like to note that due to the structure of this example, the worst-case-based synthesis problem can be solved without having to deal with the belief space at all. As the environment cannot be in state c_2, c_4, c_6 or c_7 when it is in mode $\mathcal{M}_1^{env}$ and these states only have transitions among themselves, once the environment transitions to one of these states, we know for sure that it can only be in mode $\mathcal{M}_2^{env}$ and cannot change its mode anymore. In states c_1, c_3 and c_5, the environment can be in either mode. Based on this structure and Remark 1, we can construct an AMDP that represents the complete system with smaller number of states than $\mathcal{M}^{\mathcal{A}}$ constructed using the method described in Section V-B. Exploiting the structure of the problem to reduce the size of AMDP is subject to future work.

VII. CONCLUSIONS AND FUTURE WORK

We took an initial step towards solving POMDPs that are subject to temporal logic specifications. In particular, we considered the problem where the system interacts with its dynamic environment. A collection of possible environment models are available to the system. Different models correspond to different modes of the environment. However, the system does not know in which mode the environment is. In addition, the environment may change its mode during an execution. Control policy synthesis was considered with respect to two different objectives: maximizing the expected

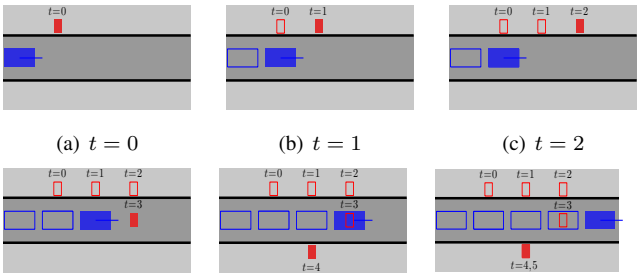


(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

(d) $t = 3$ (e) $t = 4$ (f) $t = 5$

(g) $t = 6$ (h) $t = 7$

Fig. 4. Expectation-based control policy. At time $0 \leq t < 3$, the vehicle applies α_1. α_2 is applied at time $3 \leq t < 7$, after which the vehicle reaches the goal and applies α_1 forever.



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

(d) $t = 3$ (e) $t = 4$ (f) $t = 5$

Fig. 5. Worst-case-based control policy. At time $0 \leq t < 2$, the vehicle applies α_1. However, the vehicle moves forward during time $0 \leq t < 1$ because of the uncertainties in the vehicle model. α_2 is applied at time $2 \leq t < 5$, after which the vehicle reaches the goal and applies α_1 forever.

probability and maximizing the worst-case probability that the system satisfies a given temporal logic specification.

Future work includes investigating methodologies to approximate the belief space with a finite set of representative points. This problem has been considered extensively in the POMDP literature. Since the value iteration used to obtain a solution to our expectation-based synthesis problem is similar to the value iteration used to solve POMDP problems where the expected reward is to be maximized, we believe that existing sampling techniques used to solve POMDP problems can be adapted to solve our problem. Another direction of research is to integrate methodologies for discrete state estimation to reduce the size of AMDP for the worst-case-based synthesis problem.

Acknowledgments

The authors gratefully acknowledge Tirthankar Bandyopadhyay for inspiring discussions.

References

[1] T. Wongpiromsarn, S. Karaman, and E. Frazzoli, "Synthesis of provably correct controllers for autonomous vehicles in urban environments," in *IEEE Intelligent Transportation Systems Conference*, 2011.

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

[3] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation,"*Automatica*, vol.45, no.2, pp. 566–571, 2009.

[4] K. G. Larsen and A. Skou, "Bisimulation through probabilistic testing," *Information and Computation*, vol. 94, no. 1, pp. 1–28, 1991.

[5] J. Desharnais, F. Laviolette, and M. Tracol, "Approximate analysis of probabilistic processes: Logic, simulation and games," in *Proceedings of the International Conference on Quantitative Evaluation of Systems*, pp. 264–273, 2008.

[6] A. Abate, A. D'Innocenzo, and M. Di Benedetto, "Approximate abstractions of stochastic hybrid systems," *IEEE Transaction on Automatic Control*, vol. 56, no. 11, pp. 2688–2694, 2011.

[7] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *IEEE Transaction on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

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

[9] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *Proc. of IEEE Conference on Decision and Control*, 2009.

[10] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2689–2696, 2010.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[12] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Hybrid Systems: Computation and Control*, 2010.

[13] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking and Abstract Interpretation*, vol. 3855 of *Lecture Notes in Computer Science*, pp. 364 – 380, Springer-Verlag, 2006.

[14] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *IFAC World Congress*, 2011.

[15] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *Proc. of IEEE Conference on Decision and Control*, 2011.

[16] D. D. Vecchio, R. M. Murray, and E. Klavins, "Discrete state estimators for systems on a lattice," *Automatica*, vol. 42, 2006.

[17] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[18] C. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of Operations Research*, vol. 12, pp. 441–450, 1987.

[19] M. Hauskrecht, "Value-function approximations for partially observable markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.

[20] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, pp. 780–798, 2005.

[21] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theoretical Computer Science*, vol. 363, pp. 182–195, 2006.

[22] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[23] E. A. Emerson, "Temporal and modal logic," *Handbook of Theoretical Computer Science (Vol. B): Formal Models and Semantics*, pp. 995–1072, 1990.

[24] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.

[25] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: an anytime algorithm for POMDPs," in *Proc. of International Joint Conference on Artificial Intelligence*, pp. 1025–1030, 2003.

[26] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, pp. 65–72, 2008.

[27] T. Wongpiromsarn and E. Frazzoli, "Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications," tech. rep., 2012. Available at `http://arxiv.org/abs/1203.1177`.