LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees [⋆]

Xu Chu (Dennis) Ding [*] **Stephen L. Smith** [**] **Calin Belta** [*] **Daniela Rus** [***]

[*] *Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA. (e-mail: {xcding; cbelta}@bu.edu)*
[**] *Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (email: stephen.smith@uwaterloo.ca).*
[***] *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, (e-mail: rus@csail.mit.edu)*

Abstract: We present a method to generate a robot control strategy that maximizes the probability to accomplish a task. The task is given as a Linear Temporal Logic (LTL) formula over a set of properties that can be satisfied at the regions of a partitioned environment. We assume that the probabilities with which the properties are satisfied at the regions are known, and the robot can determine the truth value of a proposition only at the current region. Motivated by several results on partitioned-based abstractions, we assume that the motion is performed on a graph. To account for noisy sensors and actuators, we assume that a control action enables several transitions with known probabilities. We show that this problem can be reduced to the problem of generating a control policy for a Markov Decision Process (MDP) such that the probability of satisfying an LTL formula over its states is maximized. We provide a complete solution for the latter problem that builds on existing results from probabilistic model checking. We include an illustrative case study.

Keywords: Robot Control, Markov Decision Processes, Formal Methods, Temporal Logic

1. INTRODUCTION

Recently there has been an increased interest in using temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) as motion specification languages for robotics [Kress-Gazit et al., 2007, Karaman and Frazzoli, 2009, Kloetzer and Belta, 2008, Wongpiromsarn et al., 2009]. Temporal logics are appealing because they provide formal, high level languages in which to describe complex missions, *e.g.,* "Reach A, then B, and then C, in this order, infinitely often. Never go to A. Don't go to B unless C or D were visited." In addition, off-the-shelf model checking algorithms [Clarke et al., 1999] and temporal logic game strategies [Piterman et al., 2006] can be used to verify the correctness of robot trajectories and to synthesize robot control strategies.

Motivated by several results on finite abstractions of control systems, in this paper we assume that the motion of the robot in the environment is modeled as a finite labeled transition system. This can be obtained by simply partitioning the environment and labeling the edges of the corresponding quotient graph according to the motion capabilities of the robot among the regions. Alternatively, the partition can be made in the state space of the robot dynamics, and the transition system is then a finite abstraction of a continuous or hybrid control system [Alur et al., 2000].

The problem of controlling a finite transition system from a temporal logic specification has received a lot of attention during recent years. All the existing works assume that the current state can be precisely determined. If the result of a control action is deterministic, control strategies from specifications given as LTL formulas can be found through an adaptation of off-the-shelf model checking algorithms [Kloetzer and Belta, 2008]. If the control is non-deterministic (an available control at a state enables one of several transitions with unknown probabilities), the control problem can be mapped to the solution of a Rabin game [Thomas, 2002], or simpler GR(1) games if the specification is restricted to fragments of LTL [Kress-Gazit et al., 2007]. If the control is probabilistic (an available control at a state enables one of several transitions with known probabilities), the transition system is a Markov Decision Process (MDP). The control problem then reduces to generating a policy (adversary) for an MDP such that the produced language satisfies a formula of a probabilistic temporal logic [Dianco and Alfaro, 1995, Kwiatkowska et al., 2004]. We have recently developed a framework for deriving an MDP control strategy from a formula in a fragment of probabilistic CTL (pCTL) [Lahijanian et al., 2010].

[⋆] This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, and NSF CNS-0834260.

In this paper, we consider motion specifications given as arbitrary LTL formulas over a set of properties that can be satisfied with given probabilities at the vertices of a graph environment. We assume that the truth values of the properties can be observed only when a vertex is reached in the environment, and the observations of these properties are independent with each other. We assume a probabilistic robot control model and that the robot can determine its current vertex precisely. Under these assumptions, we develop an algorithm to generate a control strategy that maximizes the probability of satisfying the specification. Our approach is based on mapping this problem to the problem of generating a control policy for a MDP such that the probability of satisfying an LTL formula is maximized. We provide a solution to this problem by drawing inspiration from probabilistic model checking. We illustrate the method by applying it to a numerical example of a robot navigating in an indoor environment.

The contribution of this work is twofold. First, we adapt existing approaches in probabilistic model checking (*e.g.* Baier and Katoen [2008], Dianco and Alfaro [1995]), and provide a complete solution to the general problem of controlling MDPs from full LTL specifications using deterministic Rabin automata. Second, we allow for non-determinism not only in the robot motion, but also in the robot's observation of properties in the environment. This allows us to model a large class of robotic problems in which the satisfaction of properties of interest can be predicted only probabilistically.

2. PRELIMINARIES

2.1 Linear Temporal Logic

We employ Linear Temporal Logic (LTL) to describe MDP control specifications. A detailed description of the syntax and semantics of LTL is beyond the scope of this paper and can be found in Baier and Katoen [2008], Clarke et al. [1999]. Roughly, an LTL formula is built up from a set of atomic propositions Π, standard Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and temporal operators $\bigcirc$ (next), $\mathcal{U}$ (until), $\Diamond$ (eventually), $\Box$ (always). The semantics of LTL formulas are given over infinite words $o = o_0 o_1 \ldots$ in 2^{Π}. A word satisfies an LTL formula ϕ if ϕ is true at the first position of the word; $\Box\phi$ means that ϕ is true at all positions of the word; $\Diamond\phi$ means that ϕ eventually becomes true in the word; $\phi_1 \,\mathcal{U} \phi_2$ means that ϕ_1 has to hold at least until ϕ_2 is true. More expressivity can be achieved by combining the above temporal and Boolean operators. We say $o \vDash \phi$ if the word o satisfies ϕ. An LTL formula can be represented by a deterministic *Rabin automaton*, which is defined as follows.

Definition 2.1. (Deterministic Rabin Automaton). A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$, where (i) Q is a finite set of states; (ii) Σ is a set of inputs (alphabet); (iii) $\delta : Q \times \Sigma \to Q$ is the transition function; (iv) $q_0 \in Q$ is the initial state; and (v) $F = \{(L_1, K_1), \ldots, (L_k, K_k)\}$ is a set of pairs where $L_i, K_i \subseteq Q$ for all $i \in \{1, \ldots, k\}$.

A run of a Rabin automaton $\mathcal{R}$, denoted by $r_{\mathcal{R}} = q_0 q_1 \ldots$, is an infinite sequence of states in $\mathcal{R}$ such that for each $i \geq 0$, $q_{i+1} \in \delta(q_i, \alpha)$ for some $\alpha \in \Sigma$. A run $r_{\mathcal{R}}$ is *accepting* if for a pair $(L, K) \in F$, $r_{\mathcal{R}}$ intersects with L only finitely many times and K infinitely many times.

For any LTL formula ϕ over Π, one can construct a DRA with input alphabet $\Sigma = 2^{\Pi}$ accepting all and only words over Π that satisfy ϕ. We refer readers to Klein and Baier [2006] and references therein for algorithms and to freely available implementations, such as Klein [2007], to translate a LTL formula over Π to a corresponding DRA.

2.2 Markov Decision Process and probability measure

Definition 2.2. (Labeled Markov Decision Process). A labeled Markov decision process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$, where (i) $\mathcal{S}$ is a finite set of states; (ii) $\mathcal{U}$ is a finite set of actions; (iii) $\mathcal{A} : \mathcal{S} \to 2^{\mathcal{U}}$ represents the set of actions enabled at state $s \in \mathcal{S}$; (iv) $\mathcal{P} : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$ is the transition probability function such that for all states $s \in \mathcal{S}$, $\sum_{s' \in \mathcal{S}} \mathcal{P}(s, u, s') = 1$ if $u \in \mathcal{A}(s) \subseteq \mathcal{U}$ and $\mathcal{P}(s, u, s') = 0$ if $u \notin \mathcal{A}(s)$; (v) $\iota : \mathcal{S} \to [0, 1]$ is the initial state distribution satisfying $\sum_{s \in \mathcal{S}} \iota(s) = 1$; (vi) Π is a set of atomic propositions; and (vii) $h : \mathcal{S} \to 2^{\Pi}$ is a labeling function.

We define an action function as a function $\mu : \mathcal{S} \to \mathcal{U}$ such that $\mu(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$. An infinite sequence of action functions $M = \{\mu_0, \mu_1, \ldots\}$ is called a *policy*. One can use a policy to resolve all nondeterministic choices in an MDP by applying the action $\mu_k(s_k)$ at each time-step k. Given an initial state s_0 such that $\iota(s_0) > 0$, an infinite sequence $r^M_{\mathcal{M}} = s_0 s_1 \ldots$ on $\mathcal{M}$ generated under a policy $M = \{\mu_0, \mu_1, \ldots\}$ is called a path on $\mathcal{M}$ if $\mathcal{P}(s_i, \mu_i(s_i), s_{i+1}) > 0$ for all i. If $\mu_i = \mu$ for all i, then we call this policy a stationary policy.

We define $\mathrm{Paths}^M_{\mathcal{M}}$ as the set of all paths of $\mathcal{M}$ under a policy M starting from any state s_0 where $\iota(s_0) > 0$. We can now define a probability measure over the set $\mathrm{Paths}^M_{\mathcal{M}}$. The definition of this measure can be found in a text in probabilistic model checking, such as Baier and Katoen [2008, Ch. 10]. This measure enables us to define the probability that an MDP $\mathcal{M}$ under a policy M satisfies an LTL formula ϕ. A path $r^M_{\mathcal{M}} = s_0 s_1 \ldots$ deterministically generates a word $o = o_0 o_1 \ldots$, where $o_i = \mathcal{L}(s_i)$ for all i. We denote $\mathcal{L}(r^M_{\mathcal{M}})$ as the word generated by $r^M_{\mathcal{M}}$. Given an LTL formula ϕ, one can show that the set $\{r^M_{\mathcal{M}} \in \mathrm{Paths}^M_{\mathcal{M}} : \mathcal{L}(r^M_{\mathcal{M}}) \vDash \phi\}$ is measurable. We define

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

as the probability of satisfying ϕ for $\mathcal{M}$ under M. See Baier and Katoen [2008] for more details about probability measures on MDPs under a policy and measurability of LTL formulas.

3. MODEL, PROBLEM FORMULATION, AND APPROACH

In this section we formalize the environment model, the robot motion model, and the robot observation model. We then formally state our problem and provide a summary of our technical approach.

3.1 *Environment, task, and robot model*

Environment model: In this paper, we consider a robot moving in a partitioned environment, which can be represented by a graph and a set of properties:

$$\mathcal{E} = (V, \delta_{\mathcal{E}}, \Pi), \tag{2}$$

where V is the set of vertices, $\delta_{\mathcal{E}} \subseteq V \times V$ is the relation modeling the set of edges, and Π is the set of properties (or atomic propositions). Such a finite representation of the environment can be obtained by using popular partition schemes, such as triangulations or rectangular grids. The set V can be considered as a set of labels for the regions in the partitioned environment, and $\delta_{\mathcal{E}}$ is the corresponding adjacency relation. In this paper we assume that there is no blocking vertex in V (*i.e.*, all vertices have at least one outgoing edge).

Task specification: The atomic propositions Π represent properties in the environment that can be true or false. We require the motion of the robot in the environment to satisfy a rich specification given as an LTL formula ϕ over Π (see Sec. 2).

Robot motion model: The motion capability of the robot in the environment is represented by a set of motion primitives U, and a function $A : V \to 2^U$ that returns the set of motion primitives available (or enabled) at a vertex $v \in V$. For example, U can be $\{\text{Turn Left}, \text{Turn Right}, \text{Go Straight}\}$ in an urban environment with roads and intersections. To model nondeterminism due to possible actuation or measurement errors, we define the transition probability function $P_m : V \times U \times V \to [0, 1]$ as the probability that after applying a motion primitive $u \in A(v)$ at vertex v, the robot moves from v to an adjacent region v' without passing through other regions. The set U corresponds to a set of feedback controllers for the robot, which can be constructed from facet reachability (see Habets and van Schuppen [2004]), and the transition probabilities can be obtained from experiments (see Lahijanian et al. [2010]). Note that this model of motion uses an underlying assumption that transition probabilities of the robot controllers do not depend on the previous history of the robot.

Robot observation model: In our earlier work [Kloetzer and Belta, 2008, Lahijanian et al., 2010], we assumed that some propositions in Π are associated with each region in the environment (*i.e.*, for each $v \in V$), and they are fixed in time. However, this assumption is restrictive and often not true in practice. For example, the robot might move to a road and find it congested; while finding parking spots, some parking spots may already be taken; or while attempting to upload data at an upload station, the upload station might be occupied. We wish to design control strategies that react to information which is observed in real-time, *e.g.* if a road is blocked, then pick another route.

Motivated by these scenarios, in this paper we consider the problem setting where observations of the properties of the environment are probabilistic. To this end, we define a probability function $P_o : V \times \Pi \to [0, 1]$. Thus, $P_o(v, \pi)$ is the probability that the atomic proposition $\pi \in \Pi$ is observed at a vertex $v \in V$ when v is visited. We assume that all observations of atomic propositions for a vertex $v \in V$ are independent and identically distributed. This is a reasonable model in situations where the time-scale of robot travel is larger than the time scale on which the proposition changes. For future work, we are pursuing more general observation models. Let $\Pi_v := \{\pi \in \Pi : P_o(v, \pi) > 0\}$ be the atomic propositions that can be observed at a vertex v. Then $Z_v = \{Z \subseteq \Pi_v : \prod_{\pi \in Z} P_o(v, \pi) \times \prod_{\pi \notin Z} (1 - P_o(v, \pi)) > 0\}$ is the set of all possible sets of observations $Z \subseteq \Pi_v$ at v.

3.2 *Problem Formulation*

Let the initial state of the robot be given as v_0. The trajectory of the robot in the environment is an infinite sequence $r = v_0 v_1, \ldots$, where $P_m(v_i, u, v_{i+1}) > 0$ for some u for all i. Given $r = v_0 v_1, \ldots$, we call v_i the state of the robot at the discrete time-step i. We denote the observed atomic propositions at time-step i as $o_i \in Z_{v_i}$ and $O(r) = o_0 o_1 \ldots$ as the word observed by r.

Our desired "reactive" control strategy is in the form of an infinite sequence $C = \{\nu_0, \nu_1, \ldots\}$ where $\nu_i : V \times 2^\Pi \to U$ and $\nu_i(v, Z)$ is defined only if $Z \in Z_v$. Furthermore, we enforce that $\nu_i(v, Z) \in A(v)$ for all v and all i. The reactive control strategy returns the control to be applied at each time-step, given the current state v and observed set of propositions Z at v. Given an initial condition v_0 and a control strategy C, we can produce a trajectory $r = v_0 v_1 \ldots$ where the control applied at time i is $\nu_i(v_i, o_i)$. We call r and $O(r) = o = o_0 o_1 \ldots$ the trajectory and the word generated under C, respectively. Note that given v_0 and a control strategy C, the resultant trajectory and its corresponding word are not unique due to nondeterminism in both motion and observation of the robot.

Now we formulate the following problem:

Problem 3.1. Given the environment represented by $\mathcal{E} = (V, \delta_{\mathcal{E}}, \Pi)$; the robot motion model U, A and P_m; the observation model P_o; and an LTL formula ϕ over Π, find a control strategy C that maximizes the probability that the word generated under C satisfies ϕ.

Note that a solution to Prob. 3.1 is generally not unique.

4. MDP CONSTRUCTION AND PROBLEM REFORMULATION

As part of our approach to solve Problem 3.1, we construct a labeled MDP (see Def. 2.2) $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$ from the environment model $\mathcal{E}$, the robot motion model U, A, P_m, and the observation model P_o as follows:

- $\mathcal{S} = \{(v, Z) \mid v \in V, Z \in Z_v\}$
- $\mathcal{U} = U$
- $\mathcal{A}((v, Z)) = A(v)$
- $\mathcal{P}((v, Z), u, (v', Z')) =$

$$P_m(v, u, v') \times \left(\prod_{\pi \in Z'} P_o(v', \pi) \times \prod_{\pi \notin Z'} (1 - P_o(v', \pi)) \right)$$

- ι is defined as $\iota(s) = \prod_{\pi \in Z} \mathcal{P}(v_0, \pi) \times \prod_{\pi \notin Z} (1 - \mathcal{P}(v_0, \pi))$ if $s = (v_0, Z)$ for any $Z \in Z_{v_0}$, and $\iota(s) = 0$ otherwise.
- $h((v, Z)) = Z$ for all $(v, Z) \in S$.

We now formulate a problem on the MDP $\mathcal{M}$. We will then show that this new problem is equivalent to Prob. 3.1.

Problem 4.1. For a given labeled MDP $\mathcal{M}$ and an LTL formula ϕ, find a policy such that $\Pr^M_{\mathcal{M}}(\phi)$ (see Eq. (1)) is maximized.

The following proposition formalizes the equivalence between the two problems, and the one-to-one correspondence between a control strategy on $\mathcal{E}$ and a policy on $\mathcal{M}$.

Proposition 4.1. A control strategy $C = \{\nu_0, \nu_1, \ldots\}$ is a solution to Problem 3.1 if and only if the policy $M = \{\mu_0, \mu_1, \ldots\}$, where

$$\mu_i\big((v_i, Z_i)\big) = \nu_i(v_i, Z_i) \quad \text{for each } i,$$

is a solution to Problem 4.1.

Proof. See Technical Report Ding et al. [2011].

Due to the above proposition, we will proceed by constructing a policy M on the MDP $\mathcal{M}$ as a solution to Prob. 4.1. We can then uniquely map M to a control strategy C in the robot environment $\mathcal{E}$ for a solution to Prob. 3.1.

5. SYNTHESIS OF CONTROL STRATEGY

In this section we provide a solution for Prob. 3.1 by synthesizing an optimal policy for Prob. 4.1. Our approach is adapted from automata-theoretic approaches in the area of probabilistic model checking (see Dianco and Alfaro [1995], Baier and Katoen [2008, Ch. 10] and references therein). Probabilistic LTL model checking finds the maximum probability that a path of a given MDP satisfies an LTL specification. We modify this method to obtain an optimal policy that achieves the maximum probability. This approach is related to the work of Courcoubetis and Yannakakis [1998], in which rewards are assigned to specifications and non-deterministic Büchi automata (NBA) are used. We do not use NBAs since a desired product MDP cannot be directly constructed from an NBA, but only from an DRA. Our method is also related to Baier et al. [2004], in which a control strategy is synthesized for an MDP where some states are under control of the environment, so that an LTL specification is guaranteed to be satisfied under all possible environment behaviors.

We proceed by converting the LTL formula ϕ to a DRA as in Def. 2.1. We denote the resulting DRA as $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ with $F = \{(L_1, K_1), \ldots, (L_k, K_k)\}$ where $L_i, K_i \subseteq Q$ for all $i = 1, \ldots, k$. We now obtain an MDP as the product of a labeled MDP $\mathcal{M}$ and a DRA $\mathcal{R}_\phi$. This product MDP allows one to find runs on $\mathcal{M}$ that generate words satisfying the acceptance condition of $\mathcal{R}_\phi$.

Definition 5.1. (Product MDP). The product MDP $\mathcal{M} \times \mathcal{R}_\phi$ between a labeled MDP $\mathcal{M} = (\mathcal{S}, \mathcal{U}, \mathcal{A}, \mathcal{P}, \iota, \Pi, h)$ and a DRA $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ is a tuple $\mathcal{M}_\mathcal{P} = (\mathcal{S}_\mathcal{P}, \mathcal{U}, \mathcal{A}_\mathcal{P}, \mathcal{P}_\mathcal{P}, \iota_\mathcal{P}, F_\mathcal{P})$, where:

- $\mathcal{S}_\mathcal{P} = \mathcal{S} \times Q$ (the Cartesian product of sets $\mathcal{S}$ and Q)
- $\mathcal{A}_\mathcal{P}((s,q)) = \mathcal{A}(s)$
- $\mathcal{P}_\mathcal{P}((s,q), u, (s',q')) =$
$$\begin{cases} \mathcal{P}(s,u,s') & \text{if } q' = \delta(q, h(s')) \\ 0 & \text{otherwise} \end{cases}$$
- $\iota_\mathcal{P}((s,q)) = \iota(s)$ if $q = \delta(q_0, h(s))$ and $\iota_\mathcal{P} = 0$ otherwise
- $F_\mathcal{P} = \{(L^\mathcal{P}_1, K^\mathcal{P}_1), \ldots, (L^\mathcal{P}_k, K^\mathcal{P}_k)\}$ where $L^\mathcal{P}_i = S \times L_i$, $K^\mathcal{P}_i = S \times K_i$ for all $i = 1, \ldots, k$.

Note that the set of actions for $\mathcal{M}_\mathcal{P}$ is the same as the one for $\mathcal{M}$. A policy $M_\mathcal{P} = \{\mu^\mathcal{P}_0, \mu^\mathcal{P}_1, \ldots\}$ on $\mathcal{M}_\mathcal{P}$ directly induces a policy $M = \{\mu_0, \mu_1, \ldots\}$ on $\mathcal{M}$ by keeping track of the state on the product MDP ($\mu^\mathcal{P}_i$ is an action function that returns an action corresponding to a state in $\mathcal{M}_\mathcal{P}$). Note that given the state of $\mathcal{M}$ at time-step i and the state of $\mathcal{M}_\mathcal{P}$ at time-step $i-1$, the state of $\mathcal{M}_\mathcal{P}$ at time-step i can be exactly determined. We can induce a policy M for $\mathcal{M}$ from a policy $M_\mathcal{P}$ for $\mathcal{M}_\mathcal{P}$ as follows:

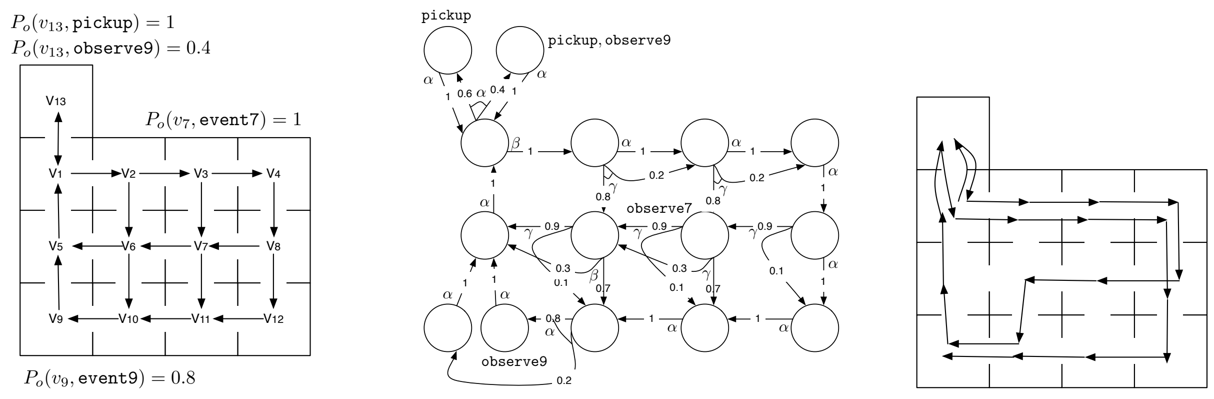
Definition 5.2. (Inducing a policy for $\mathcal{M}$ from $\mathcal{M}_\mathcal{P}$). If the state of $\mathcal{M}_\mathcal{P}$ at time-step i is (s_i, q_i), then the policy $M = \{\mu_0, \mu_1, \ldots\}$ induced from $M_\mathcal{P} = \{\mu^\mathcal{P}_0, \mu^\mathcal{P}_1 \ldots\}$ can be obtained by setting $\mu_i(s_i) = \mu^\mathcal{P}_i((s_i, q_i))$ for all i. We say a path $r^{M_\mathcal{P}}_{\mathcal{M}_\mathcal{P}}$ is accepting if and only if it satisfies the Rabin acceptance condition with $F_\mathcal{P}$ as the accepting states pairs.

From probabilistic model checking, the product MDP is constructed so that given a path $r^{M_\mathcal{P}}_{\mathcal{M}_\mathcal{P}} = (s_0, q_0)(s_1, q_1)\ldots$, the path $s_0 s_1 \ldots$ on $\mathcal{M}$ satisfies ϕ if and only if $r^{M_\mathcal{P}}_{\mathcal{M}_\mathcal{P}}$ is accepting. We can then obtain a set of accepting maximum end components (AMECs) of $\mathcal{M}_\mathcal{P}$. An AMEC for $\mathcal{M}_\mathcal{P}$ consists of a set of states $\overline{\mathcal{S}_\mathcal{P}} \subseteq \mathcal{S}_\mathcal{P}$ and function $\overline{\mathcal{A}_\mathcal{P}}$ such that $\emptyset \neq \overline{\mathcal{A}_\mathcal{P}}((s,q)) \subseteq \mathcal{A}_\mathcal{P}((s,q))$ for all $(s,q) \in \overline{\mathcal{S}_\mathcal{P}}$, with the property that, by taking actions enabled by $\overline{\mathcal{A}_\mathcal{P}}$, all states in $\overline{\mathcal{S}_\mathcal{P}}$ can reach every other state in $\overline{\mathcal{S}_\mathcal{P}}$ and can not reach any state outside of $\overline{\mathcal{S}_\mathcal{P}}$. Furthermore, it contains no state in $L^\mathcal{P}_i$ and at least one state in $K^\mathcal{P}_i$ and is not properly contained in another such component. A procedure to obtain all AMECs of an MDP is outlined in Baier and Katoen [2008]. The maximum probability of satisfying the LTL formula ϕ for $\mathcal{M}$ is the same as the maximum probability of reaching any AMEC of $\mathcal{M}_\mathcal{P}$. Once an AMEC $(\overline{\mathcal{S}_\mathcal{P}}, \overline{\mathcal{A}_\mathcal{P}})$ is reached, all states in $\overline{\mathcal{S}_\mathcal{P}}$ are reached infinitely often (and ϕ satisfied) with probability 1 under a policy that using all actions in $\overline{\mathcal{A}_\mathcal{P}}$ infinitely often.

Our desired policy $M^\star_\mathcal{P}$ that maximizes the probability of satisfying ϕ on the product MDP is the policy maximizing the probability of reaching a set of states, which is the union of all AMECs of all accepting state pairs in $F_\mathcal{P}$, if the state is not already in an AMEC. A policy maximizing the probability of reaching a set of states on an MDP can be found by the solution of a linear program (see *e.g.* Ding et al. [2011]). If the state of the MDP is inside an AMEC, the optimal policy is to use all enabled actions in the AMEC infinitely often in a round-robin fashion. The solution to Prob. 4.1 is then the policy $M^\star$ on $\mathcal{M}$ induced by $M^\star_\mathcal{P}$. The desired reactive control strategy $C^\star$ as a solution to Prob. 3.1 can finally be obtained as the control strategy corresponding to $M^\star$ (see Prop. 4.1). Our overall approach is summarized in Alg. 1.

5.1 Complexity

The complexity of our proposed algorithm is dictated by the size of the generated MDPs. We use $|\cdot|$ to denote cardinality of a set. The number of states in $\mathcal{M}$ is $|\mathcal{S}| = \sum_{v \in V} |Z_v|$. Hence, in the worst case where all propositions $\pi \in \Pi$ can be observed with positive but less than 1 probability at all vertices $v \in V$, $|\mathcal{S}| = 2^{|\Pi|}$.



$P_o(v_{13}, \texttt{pickup}) = 1$
$P_o(v_{13}, \texttt{observe9}) = 0.4$
V13
$P_o(v_7, \texttt{event7}) = 1$
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
$P_o(v_9, \texttt{event9}) = 0.8$
pickup
pickup, observe9
observe7
observe9
(a)
(b)
(c)

Fig. 1. **(a):** Environment for a numerical example of the proposed approach. We assume that the set of motion primitive is $U = \{\alpha, \beta, \gamma\}$. We define the enabling function A so that the motion primitive α is enabled at all vertices, β is enabled at vertices v_1, v_6 and v_7, and γ is enabled at vertices v_2, v_3, v_6, v_7 and v_8. **(b):** MDP $\mathcal{M}$ generated from the environment with given U, A, P_o and P_m. **(c):** A sample path of the robot with the optimal control strategy. The word observed by the sample path is `pickup`, `event7`, `event9`, {`pickup`, `observe9`}, `event9`, ….

Algorithm 1 Generating the optimal control strategy $C^\star$ given $\mathcal{E}$, U, A, P_m, P_o and ϕ

```
1: Generate the MDP $\mathcal{M}$ from the environment model $\mathcal{E}$, the motion primitives $U$, the actions $A$, the motion model $P_m$ and the observation model $P_o$.
2: Translate the LTL formula $\phi$ to a DRA $\mathcal{R}_\phi$.
3: Generate the product MDP $\mathcal{M}_\mathcal{P} = \mathcal{M} \times \mathcal{R}_\phi$.
4: Find all AMECs $(\overline{\mathcal{S}_\mathcal{P}}, \overline{\mathcal{A}_\mathcal{P}})$ for all pairs $(L_i^\mathcal{P}, K_i^\mathcal{P}) \in F_\mathcal{P}$, and find the union $B_\mathcal{P}$ of all $\overline{\mathcal{S}_\mathcal{P}}$'s.
5: Find the stationary policy $\{\mu_\mathcal{P}^\star, \mu_\mathcal{P}^\star, \ldots\}$ maximizing the probability of reaching $B_\mathcal{P}$.
6: Generate the policy $M_\mathcal{P}^\star = \{\mu_0^\mathcal{P}, \mu_1^\mathcal{P}, \ldots\}$ as follows: $\mu_i^\mathcal{P}(p) = \mu_\mathcal{P}^\star(p)$ if $p \in \mathcal{S}_\mathcal{P} \setminus B_\mathcal{P}$. Otherwise, $p$ is in at least one AMEC. Assuming it is $(\overline{\mathcal{S}_\mathcal{P}}, \overline{\mathcal{A}_\mathcal{P}})$ and $\overline{\mathcal{A}_\mathcal{P}}(p) = \{u_1, u_2, \ldots, u_m\}$, then $\mu_i^\mathcal{P}(p) = u_j$ where $j = i \bmod m$.
7: Generate policy $M^\star = \{\mu_0, \mu_1, \ldots\}$ induced by $M_\mathcal{P}^\star$.
8: Generate the optimal control strategy $C^\star = \{\nu_0, \nu_1, \ldots\}$ from $M^\star$ using Prop. 4.1.
```

In practice, the number of propositions that can be non-deterministically observed at a vertex is small. The size of the DRA $|Q|$ is in worst case, doubly exponential with respect to $|\Pi|$. However, empirical studies such as Klein and Baier [2006] have shown that in practice, the sizes of the DRAs for many LTL formulas are exponential or lower with respect to $|\Pi|$. In robot control applications, since properties in the environment are typically assigned scarcely (meaning that each region of the environment is usually assigned a small number of properties comparing to $|\Pi|$), the size of DRA can be reduced much further by removing transitions in the DRA with inputs that can never appear in the environment and unreachable states. The size of the product MDP is $|\mathcal{M}| \times |Q|$. The complexity for the algorithm generating AMECs is at most quadratic in the size of $\mathcal{M}_\mathcal{P}$ (see Baier and Katoen [2008]), and the complexity for finding the optimal policy from a linear program is polynomial in the size of $\mathcal{M}_\mathcal{P}$. Thus, overall, our algorithm is polynomial in the size of $\mathcal{M}_\mathcal{P}$.

6. EXAMPLE

The computational framework developed in this paper is implemented in MATLAB, and here we provide an example as a case study. Note that we did not use probabilistic model-checkers such as PRISM (see Kwiatkowska et al. [2004]) because they return only maximal probabilities and not optimal policies.

Consider a robot navigating in an indoor environment as shown in Fig. 1a. Each region of the environment is represented by a vertex v_i, and the arrows represent allowable transitions between regions. In this case study, we choose the motion primitives arbitrarily (see the caption of Fig. 1a). In practice, they can either correspond to low level control actions such as “turn left”, “turn right” and “go straight”, or high level commands such as “go from region 1 to region 2”, which can then be achieved by a sequence of low level control actions.

The goal of the robot is to perform a persistent surveillance mission on regions v_7 and v_9, described as follows: The robot can pickup (or receive) a surveillance task at region v_{13}. With probability 0.4 the robot receives the task denoted `observe9`. Otherwise, the task is `observe7`. The task `observe7` (or `observe9`) is completed by traveling to region v_7 (or v_9), and observing some specified event. In region v_7, the robot observes the event (`event7`) with probability 1. In region v_9, each time the robot enters the region, there is a probability of 0.8 that it observes the event (`event9`). Thus, the robot may have to visit v_9 multiple times before observing `event9`. Once the robot observes the required event, it must return to v_{13} and pickup a new task. This mission can be specified with the set of atomic propositions {`pickup`, `observe9`, `event7`, `event9`} (the task `observe7` can be written as $\neg$`observe9`). The propositions `pickup` and `observe7` are assigned to v_{13}, with $P_o(v_{13}, \texttt{pickup}) = 1$ and $P_o(v_{13}, \texttt{observe9}) = 0.4$. The proposition `event7` is assigned to v_7 with $P_o(v_7, \texttt{event7}) = 1$ and `event9` is assigned to v_9 with $P_o(v_9, \texttt{event9}) = 0.8$.

The mission specification can be written as:

$$\phi = \Box\Diamond\texttt{pickup}\wedge$$
$$\Box\,(\texttt{pickup} \wedge \neg\texttt{observe9} \Rightarrow \bigcirc(\neg\texttt{pickup}\,\mathcal{U}\texttt{event7}))$$
$$\wedge\,\Box\,(\texttt{pickup} \wedge \texttt{observe9} \Rightarrow \bigcirc(\neg\texttt{pickup}\,\mathcal{U}\texttt{event9}))\,.$$

The first line of ϕ, $\Box\Diamond\texttt{pickup}$, enforces that the robot must repeatedly pick up tasks. The second line pertains to task observe7 and third line pertains to task observe9. These two lines ensure that a new task cannot be picked up until the current task is completed (*i.e.*, the desired event is observed). Note that if event9 is observed after observing event7, then the formula ϕ is not violated (and similarly if event7 is observed after observing event9).

The MDP $\mathcal{M}$ generated from the environment is shown in Fig. 1b. For this example, we have arbitrarily chosen values for the probability transition function P_m. In practice, probabilities of transition under actuation and measurement errors can be obtained from experiments or accurate simulations (see Lahijanian et al. [2010]). The number of states in the MDP $\mathcal{M}$ is $|S| = 15$. We generated the deterministic Rabin automaton $\mathcal{R}_\phi$ using the ltl2dstar tool (see Klein [2007]). The number of states $|Q|$ is 52. Thus, the product MDP $\mathcal{M}_\mathcal{P}$ has 780 states. For the DRA generated, there is only one set in F, *i.e.*, $F = \{(L, K)\}$, with 1 state in L and 18 states in K. Thus, the number of states in $L^\mathcal{P}$ is 15 and the number of states in $K^\mathcal{P}$ is 270. There is one AMEC in $\mathcal{M}_\mathcal{P}$, and it contains 17 states.

Using the implementation of Alg. 1 we computed the maximum probability of satisfying the specification from the initial state and the optimal control strategy. The Algorithm ran in approximately 7 seconds on a MacBook Pro computer with a 2.5 GHz dual core processor. For this example the maximum probability is 1, implying that the corresponding optimal control strategy almost surely satisfies ϕ. To illustrate the control strategy, a sample execution is shown in Fig. 1c.

7. CONCLUSIONS AND FINAL REMARKS

We presented a method to generate a robot control strategy that maximizes the probability to accomplish a task. The robot motion in the environment was modeled as a graph and the task was given as a Linear Temporal Logic (LTL) formula over a set of properties that can be satisfied at the vertices with some probability. We allowed for noisy sensors and actuators by assuming that a control action enables several transitions with known probabilities and reduced this problem to one of generating a control policy for a Markov Decision Process such that the probability of reaching some of its states is maximized. We then provided a complete solution to this problem adapting existing probabilistic model checking approaches.

We are currently pursuing several future directions. We are looking at proposition observation models that are not independently distributed, *i.e.*, when the current truth value of the proposition gives information about the future truth value. We are also looking at methods for optimizing the robot control strategy for a suitable cost function when costs are assigned to actions of an MDP.

REFERENCES

R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88:971–984, 2000.

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

C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski. Controller synthesis for probabilistic systems. In *Proceedings of IFIP TCS*. 2004.

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

C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. *IEEE Transactions on Automatic Control*, 43(10):1399–1418, 1998.

A. Dianco and L. De Alfaro. Model checking of probabilistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

X. C. Ding, S. L. Smith, C. Belta, and D. Rus. LTL control in uncertain environments with probabilistic satisfaction guarantees, April 2011. available at http://arxiv.org/abs/1104.1159.

L.C.G.J.M. Habets and J.H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40(1):21–35, 2004.

S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *IEEE Conf. on Decision and Control*, pages 2222 – 2229, Shanghai, China, 2009.

J. Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/, 2007.

J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theoretical Computer Science*, 363(2):182–195, 2006.

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

H. Kress-Gazit, G. Fainekos, and G. J. Pappas. Where's Waldo? Sensor-based temporal logic motion planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 3116–3121, Rome, Italy, 2007.

M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *IEEE Int. Conf. on Robotics and Automation*, pages 3227 – 3232, Anchorage, AK, 2010.

N. Piterman, A. Pnueli, and Y. Saar. Synthesis of reactive(1) designs. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 364–380, Charleston, SC, 2006.

W. Thomas. Infinite games and verification. In E. Brinksma and K. Larsen, editors, *Computer Aided Verification*, volume 2404 of *LNCS*, pages 58–65. Springer, 2002.

T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning for dynamical systems. In *IEEE Conf. on Decision and Control*, pages 5997–6004, Shanghai, China, 2009.