Control of Markov Decision Processes from PCTL specifications

M. Lahijanian, S. B. Andersson, and C. Belta

Abstract— **We address the problem of controlling a Markov Decision Process (MDP) such that the probability of satisfying a temporal logic specification over a set of properties associated to its states is maximized. We focus on specifications given as formulas of Probabilistic Computation Tree Logic (PCTL) and show that controllers can be synthesized by adapting existing PCTL model checking algorithms. We illustrate the approach by applying it to the automatic deployment of a mobile robot in an indoor-like environment with respect to a PCTL specification.**

I. Introduction

Markov decision processes (MDPs) offer a mathematical framework for modeling systems with stochastic dynamics. These models provide an effective means for describing processes in which sequential decision making is involved. Applications can be found across an expansive array of fields, including economics, biology, and engineering. In general, the "solution" to an MDP is a control policy which minimizes a particular cost defined with respect to the states in the MDP. While powerful, there are many tasks and goals which could be considered for an MDP that cannot easily be described in terms of a cost function.

A more general, and perhaps more natural, approach to describing a specification for a given model can be found in the field of verification of stochastic systems. Under these schemes, a temporal logic, such as probabilistic Linear Temporal Logic (pLTL) [1] and Probabilistic Computation Tree Logic (PCTL) [2], are used to describe a property over the system. These properties are typically expressed over the states of an MDP and may involve temporal conditions. Examples specifications include "deadlock occurs with probability at most 0.01" and "find the maximum probability of reaching an unstable state within k steps". Model checking algorithms are then used to calculate the probability that the MDP will satisfy the given specification [3]–[7].

In the area of motion planning and robotics, recent works have suggested the use of such temporal logics as motion specification languages [8]–[12]. Their use enables increased expressivity over "classical" methods involving only state-to-state transfers. Algorithms inspired from model checking [13], [14] or temporal logic games [15] are used to find motion plans and control strategies from such specifications. Under these schemes, the system is first abstracted to a transition system. Most existing methods proceed based on two main assumptions. First, that the transition system is either purely deterministic (that is, each control action enables a unique transition) or purely nondeterministic (that is, each control action can enable multiple transitions with no information as to their likelihoods) [16]. Second, the current state of the system is known precisely. In realistic applications, however, noisy sensors and actuators can cause both of these assumptions to fail.

In order to develop an approach in which the system noise is explicitly considered, in an earlier work we considered an MDP model of a system and task specifications given in a small segment of PCTL formulas, namely those containing only a single instance of a particular temporal operator ("until") [17]. In this work, we develop a strategy for controlling an MDP with respect to a specification given in the full range of PCTL formulas. When applied to robotic systems, our approach provides a framework for robotic control from temporal logic specifications with probabilistic guarantees. As a result, our approach will automatically determine a control strategy that maximizes the probability of satisfy a rich specification as well as the corresponding probability. Examples of complex missions possible include "Eventually reach A and then B with probability greater than 0.9 while always avoiding the regions from which the probabilities of converging to D is greater than 0.2".

While the building blocks of our control synthesis algorithm are based on an adaptation of existing PCTL model checking algorithms [6], the synthesis approach to PCTL formulas with more than one temporal operator and the framework are, to the best of our knowledge, novel and quite general. In short, given a specification as a PCTL formula, the algorithm returns the maximum satisfaction probability and the corresponding control strategy. Our algorithm uses sub-algorithms corresponding to each temporal operator (including the one presented in [17]) as building blocks for construction of a control strategy from a formula with multiple temporal operators. The most computationally expensive sub-algorithm requires solving a linear programming problem.

To illustrate the method, we deployed a robot from PCTL specifications by using our Robotic InDoor Environment (RIDE) simulator [18]. This simulator mimics the motion of an iRobot iCreate platform equipped with a laptop, RFID reader, and laser range finder moving autonomously through corridors and intersections. It includes models of sensor and actuator noise which has been validated against the physical system [17]. Through the simulator, we generated the MDP model and tested the produced control strategies.

The authors are with the Department of Mechanical Engineering, Boston University, MA, USA, E-mail: {morteza,sanderss,cbelta}@bu.edu.

This work is partially supported at Boston University by the NSF under grants CNS-0834260 and CMMI-0928776, the ARO under grant W911NF-09-1-0088, the AFOSR under grant FA9550-09-1-0209, and the ONR MURI under grant N00014-09-1051.

M. Lahijanian is the corresponding author.

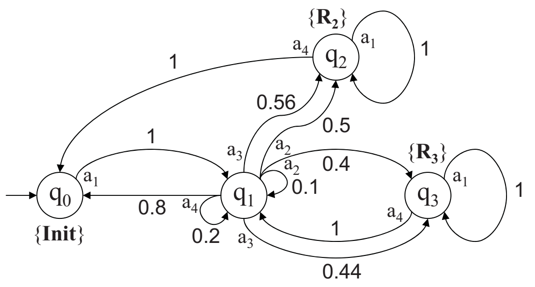

Fig. 1. A four-state MDP.

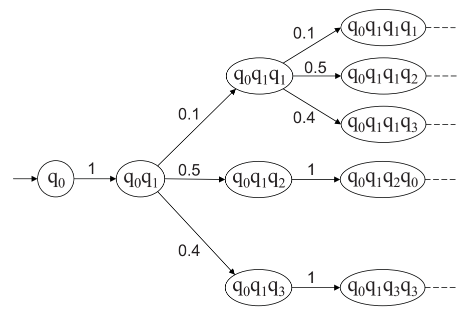

Fig. 2. Fragment of DTMCs $\mathcal{D}_{\mu_1}$ for control policy μ_1.

The remainder of the paper is organized as follows. In Sec. II, we formally define MDP, probability measure over paths of MDP, and PCTL. In Sec. III, we formulate the problem and state our approach. The MDP control synthesis from PCTL formulas and the issues of conservatism and complexity of the algorithms are discussed in Sec. IV. The results of the simulation case studies are included in Sec. V. The paper concludes with final remarks in Sec. VI.

II. Preliminaries

A. Markov Decision Process

Given a set Q, let $|Q|$ and 2^Q denote its cardinality and power set, respectively.

Definition 1 (MDP): An MDP is a tuple $\mathcal{M} = (Q, q_0, Act, Steps, \Pi, L)$ where:

- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- Act is a set of actions;
- $Steps : Q \rightarrow 2^{Act \times \Sigma(Q)}$ is a transition probability function where $\Sigma(Q)$ is the set of all discrete probability distributions over the set Q;
- Π is a finite set of atomic propositions;
- $L : Q \rightarrow 2^{\Pi}$ is a labeling function assigning to each state possibly several elements of Π;

The set of actions available at $q \in Q$ is denoted by $\mathcal{A}(q)$. The function $Steps$ is often represented as a matrix with $|Q|$ columns and $\sum_{i=0}^{|Q|-1} |\mathcal{A}(q_i)|$ rows. For each action $a \in \mathcal{A}(q_i)$, we denote the probability of transitioning from state q_i to state q_j under the action a as $\sigma_a^{q_i}(q_j)$ and the corresponding probability distribution function as $\sigma_a^{q_i}$. Each $\sigma_a^{q_i}$ corresponds to one row in the matrix representation of $Steps$.

To illustrate these definitions, a simple MDP is shown in Fig. 1. The actions available at each state are $\mathcal{A}(q_0) = \{a_1\}$, $\mathcal{A}(q_1) = \{a_2, a_3, a_4\}$, and $\mathcal{A}(q_2) = \mathcal{A}(q_3) = \{a_1, a_4\}$. The labels are $L(q_0) = \{\textbf{Init}\}$, $L(q_2) = \{\textbf{R}_2\}$, and $L(q_3) = \{\textbf{R}_3\}$. The matrix representation of $Steps$ is given by

$$Steps = \begin{array}{c} q_0;a_1 \\ \hline q_1;a_2 \\ q_1;a_3 \\ q_1;a_4 \\ \hline q_2;a_1 \\ q_2;a_4 \\ \hline q_3;a_1 \\ q_3;a_4 \end{array} \left(\begin{array}{cccc} 0 & 1 & 0 & 0 \\ \hline 0 & 0.1 & 0.5 & 0.4 \\ 0 & 0 & 0.56 & 0.44 \\ 0.8 & 0.2 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{array} \right).$$

B. Paths, Control Policies, and Probabilistic Measures

A path ω through an MDP is a sequence of states $\omega = q_0 \xrightarrow{(a_0, \sigma_{a_0}^{q_0}(q_1))} q_1 \xrightarrow{(a_1, \sigma_{a_1}^{q_1}(q_2))} \ldots q_i \xrightarrow{(a_i, \sigma_{a_i}^{q_i}(q_{i+1}))} q_{i+1} \ldots$ where each transition is induced by a choice of action at the current step i. We denote the set of all finite paths by $Path^{fin}$ and of infinite paths by $Path$.

A *control policy* is a function $\mu : Path^{fin} \rightarrow Act$. That is, for every finite path, a policy specifies the next action to be applied. If a control policy depends only on the last state of ω^{fin}, it is called a *stationary policy*. Under a policy μ, an MDP becomes a Markov chain, denoted $\mathcal{D}_{\mu}$. Let $Path_{\mu} \subseteq Path$ and $Path_{\mu}^{fin} \subseteq Path^{fin}$ denote the set of infinite and finite paths that can be produced under μ. Because there is a one-to-one mapping between $Path_{\mu}$ and the set of paths of $\mathcal{D}_{\mu}$, the Markov chain induces a probability measure over $Path_{\mu}$ as follows.

First, define a measure $Prob_{\mu}^{fin}$ over the set of finite paths by setting the probability of $\omega^{fin} \in Path_{\mu}^{fin}$ equal to the product of the corresponding transition probabilities in $\mathcal{D}_{\mu}$. Then, define $C(\omega^{fin})$ as the set of all (infinite) paths $\omega \in Path_{\mu}$ with the prefix ω^{fin}. The probability measure on the smallest σ-algebra over $Path_{\mu}$ containing $C(\omega^{fin})$ for all $\omega^{fin} \subset Path_{\mu}^{fin}$ is the unique measure satisfying

$$Prob_{\mu}(C(\omega^{fin})) = Prob_{\mu}^{fin}(\omega^{fin}) \, \forall \omega^{fin} \in Path_{\mu}^{fin}. \quad (1)$$

To illustrate this measure, consider the MDP shown in Fig. 1 and the stationary control policy defined by the mapping

$$\mu_1(\cdots q_0) = a_1, \, \mu_1(\cdots q_1) = a_2,$$
$$\mu_1(\cdots q_2) = a_4, \, \mu_1(\cdots q_3) = a_1,$$

where $\cdots q_i$ denotes any finite path terminating in q_i. The initial fragment of the resulting Markov chain is shown in Fig. 2. From this fragment it is easy to see that the probability of the finite path $q_0q_1q_2$ is $Prob_{\mu_1}(q_0q_1q_2) = 0.5$. Under μ_1, the set of all infinite paths with this prefix is

$$C(q_0q_1q_2) = \{\overline{q_0q_1q_2}, \, q_0q_1q_2q_0\overline{q_1}, \, q_0q_1q_2q_0q_1\overline{q_3}, \ldots\}$$

where the sequence under the over-line is repeated infinitely. According to (1), we have that $Prob_{\mu_1}(C(q_0q_1q_2)) = Prob_{\mu_1}(q_0q_1q_2) = 0.5$.

C. Probabilistic Computation Tree Logic (PCTL)

We use PCTL [6] , a probabilistic extension of CTL that includes a probabilistic operator $\mathcal{P}$, to write specifications of MDP.

Definition 2 (Syntax of PCTL): PCTL formulas are state formulas, which can be recursively defined as follows:

$$\phi ::= true \mid \pi \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\bowtie p}[\psi] \quad \text{state formulas}$$
$$\psi ::= X\phi \mid \phi\,\mathcal{U}^{\leq k}\,\phi \mid \phi\,\mathcal{U}\,\phi \quad \text{path formulas}$$

where $\pi \in \Pi$ is an atomic proposition, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, and $k \in \mathbb{N}$. State formulas ϕ are evaluated over states of MDP while path formulas ψ are assessed over paths and only allowed as the parameter of the $\mathcal{P}_{\bowtie p}[\psi]$ operator. Intuitively, a state q satisfies $\mathcal{P}_{\bowtie p}[\psi]$ if the probability of taking a path from q satisfying ψ is in the range $\bowtie p$. Temporal logic operators X ("next"), $\mathcal{U}^{\leq k}$ ("bounded until"), and $\mathcal{U}$ ("until") are allowed in path formulas.

Definition 3 (Semantics of PCTL): For any state $q \in Q$, the satisfaction relation $\models$ is defined inductively as follows:

- $q \models true$ for all $q \in Q$,
- $q \models \pi \Leftrightarrow \pi \in L(q)$,
- $q \models (\phi_1 \wedge \phi_2) \Leftrightarrow q \models \phi_1 \wedge q \models \phi_2$,
- $q \models \neg\phi \Leftrightarrow q \not\models \phi$,
- $q \models \mathcal{P}_{\bowtie p}[\psi] \Leftrightarrow p^q_\mu \bowtie p$

where p^q_μ is the probability of all the infinite paths that start from q and satisfy ψ under policy μ. Moreover, for any path $\omega \in Path$:

- $\omega \models X\phi \Leftrightarrow \omega(1) \models \phi$;
- $\omega \models \phi_1\mathcal{U}^{\leq k}\phi_2 \Leftrightarrow \exists i \leq k, \omega(i) \models \phi_2 \wedge \omega(j) \models \phi_1 \forall j < i$;
- $\omega \models \phi_1\,\mathcal{U}\,\phi_2 \Leftrightarrow \exists\, k \geq 0, \omega \models \phi_1\,\mathcal{U}^{\leq k}\,\phi_2$.

III. Problem Formulation and Approach

In this paper, we consider the following problem:

Problem 1: Given a Markov decision process model $\mathcal{M}$ and a PCTL specification formula ϕ over Π, find a control policy that maximizes the probability of satisfying ϕ.

Our control synthesis algorithm takes a PCTL formula ϕ and an MDP $\mathcal{M}$, and returns both the optimal probability of satisfying ϕ and the corresponding control policy. The basic algorithm proceeds by constructing the parse tree for ϕ and treating each operator in the formula separately. The method of control synthesis for each temporal operator is presented in Sec. IV-A, IV-B, and IV-C. These algorithms are inspired by model checking [6] with a few modifications such as finding all the satisfying actions for the "next" operator and producing a stationary policy for the "bounded until" operator. Moreover, we make the connection between these algorithms and the Maximum Reachability Probability problem [19]. In Sec. IV-D, we show how to construct a control strategy from a PCTL formula with nested $\mathcal{P}$-operators using the algorithms shown in Sec. IV-A, IV-B, and IV-C. It should be noted that in general we are interested in finding the control policy that produces the maximum/minimum probability of satisfying the given specification. Such PCTL formulas have the form $\mathcal{P}_{max=?}[\psi]$ and $\mathcal{P}_{min=?}[\psi]$. For PCTL formulas of the form $\mathcal{P}_{\bowtie p}[\psi]$, we still use the algorithms that return optimal policies (with the exception of the case discussed in Sec. IV-D). For these formulas, we first find the control policy μ and then check whether $p^q_\mu(\phi)$ satisfies the bound $\bowtie p$. For the case $\bowtie \in \{>, \geq\}$, we use the policy that maximizes the probability of satisfaction. Similarly, we determine the policy corresponding to minimum probability when $\bowtie \in \{<, \leq\}$.

IV. PCTL Control Synthesis

A. Next Operator

For the "next" temporal operator, we present two algorithms. One finds the optimal control strategy, and the other determines all the satisfying policies. For PCTL formulas that include only one $\mathcal{P}$-operator, the optimal control strategy algorithm is always used. For nested $\mathcal{P}$-operator formulas, both algorithms are used. This becomes clear in Sec. IV-D.

1) Next (Optimal) - $\phi = \mathcal{P}_{max=?}[X\phi_1]$: For this operator, we need to determine the action that produces the maximum probability of satisfying $X\phi_1$ at each state. Thus, we only need to consider the immediate transitions at each state. Therefore, the problem reduces to the following:

$$x^*_{q_i} = \max_{a \in \mathcal{A}(q_i)} \sum_{q_j \in Sat(\phi_1)} \sigma_a^{q_i}(q_j),$$
$$\mu^*(q_i) = \arg \max_{a \in \mathcal{A}(q_i)} \sum_{q_j \in Sat(\phi_1)} \sigma_a^{q_i}(q_j),$$

where $x^*_{q_i}$ denotes the optimal probability of satisfying ϕ at state $q_i \in Q$, $Sat(\phi_1)$ is the set of states that satisfy ϕ_1, and μ^* represents the optimal policy.

To solve the above maximization problem, we define a state-indexed vector $\overline{\phi_1}$ with entries $\overline{\phi_1}(q_i)$ equal to 1 if $q_i \models \phi_1$ and 0 otherwise. To compute the maximum probability, first, the matrix $Steps$ is multiplied by $\overline{\phi_1}$. The result is a vector whose entries are the probability of satisfying $X\phi_1$ where each row corresponds to a state-action pair. Then, the maximization operation is performed on this vector which selects the maximum probabilities and the corresponding actions at each state. The resulting control strategy is stationary, and the complexity of achieving it is one vector-matrix multiplication followed by a one dimensional search.

To demonstrate this method, consider the MDP in Fig. 1 and the formula $\phi = \mathcal{P}_{max=?}[X(\neg\mathbf{R_3})]$. The property $(\neg\mathbf{R_3})$ is satisfied at states q_0, q_1, and q_2; thus, $\overline{\neg\mathbf{R_3}} = (1\;1\;1\;0)^T$. Then, $Steps \cdot \overline{\neg\mathbf{R_3}} =$

$$\begin{array}{c} q_0;a_1 \\ \hline q_1;a_2 \\ q_1;a_3 \\ q_1;a_4 \\ \hline q_2;a_1 \\ q_2;a_4 \\ \hline q_3;a_1 \\ q_3;a_4 \end{array} \left(\begin{array}{cccc} 0 & 1 & 0 & 0 \\ \hline 0 & 0.1 & 0.5 & 0.4 \\ 0 & 0 & 0.56 & 0.44 \\ 0.8 & 0.2 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{array}\right) \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \left(\begin{array}{c} 1 \\ \hline 0.6 \\ 0.56 \\ 1 \\ \hline 1 \\ 1 \\ \hline 0 \\ 1 \end{array}\right).$$

Thus, $x_{q_i} = 1$ for $i = 0, \ldots, 3$ and the optimal stationary policy is $\mu^*(q_0) = a_1$, $\mu^*(q_1) = a_4$, $\mu^*(q_2) = a_1$ or a_4, and $\mu^*(q_3) = a_4$.

2) Next (All) - $\mathcal{P}_{\bowtie p}[X\phi_1]$: Here, we are interested in finding all the policies that satisfy the formula. The algorithm is the same as the one for Optimal Next (Sec. IV-A.1) up to the maximization step. After obtaining the vector $Steps \cdot \overline{\phi_1}$, which includes the probabilities of satisfying $X\phi_1$ for each state-action pair, we eliminate the state-action pairs whose probabilities are not in the range of $\bowtie p$. This operation determines all the states, actions, and their corresponding probabilities that satisfy $\mathcal{P}_{\bowtie p}[X\phi_1]$. It should be noted that this algorithm is only used in nested formulas (Sec. IV-D).

To illustrate this algorithm, consider the example in Sec. IV-A.1 with the formula $\phi = \mathcal{P}_{\geq 0.6}[X(\neg R_3)]$. All satisfying actions at states q_0, q_1, q_2, and q_3 are $\{a_1\}$, $\{a_2, a_4\}$, $\{a_1, a_4\}$, and $\{a_4\}$ respectively.

B. Bounded Until Operator

For this operator, we also introduce two algorithms: optimal and stationary. The optimal algorithm produces a history dependent control policy. The stationary algorithm results in a stationary policy and is used only for nested formulas.

1) Bounded Until (Optimal) - $\phi = \mathcal{P}_{max=?}[\phi_1 \mathcal{U}^{\leq k}\phi_2]$: To find the probabilities $p^q_{max}(\phi_1 \mathcal{U}^{\leq k}\phi_2)$, we first group the MDP states into three subsets: states that always satisfy the specification Q^{yes}, states that never satisfy the specification Q^{no}, and the remaining states $Q^?$.

$$\begin{aligned} Q^{yes} &= Sat(\phi_2), \\ Q^{no} &= Q \setminus (Sat(\phi_1) \cup Sat(\phi_2)), \\ Q^? &= Q \setminus (Q^{yes} \cup Q^{no}). \end{aligned}$$

Trivially, the probabilities of the states in Q^{yes} and in Q^{no} are 1 and 0 respectively. The probabilities for the remaining states $q_i \in Q^?$ are defined recursively. If $k = 0$, then $p^{q_i}_{max}(\phi_1 \mathcal{U}^{\leq k}\phi_2) = 0 \; \forall q_i \in Q^?$. For $k > 0$,

$$x^k_{q_i} = \max_{a \in \mathcal{A}(q_i)} \sum_{q_j \in Q^?} \sigma^{q_i}_a(q_j) x^{k-1}_{q_i} + \sum_{q_j \in Q^{yes}} \sigma^{q_i}_a(q_j) \, \forall q_i \in Q^?$$

$$\mu^{*^k}(q_i) = \arg \max_{a \in \mathcal{A}(q_i)} \sum_{q_j \in Q^?} \sigma^{q_i}_a(q_j) x^{k-1}_{q_i} + \sum_{q_j \in Q^{yes}} \sigma^{q_i}_a(q_j) \quad \forall q_i \in Q^?,$$

where $x^k_{q_i}$ and $\mu^{*^k}(q_i)$ denote the probability of satisfying ϕ and the corresponding optimal action at state $q_i \in Q^?$ at time step k respectively.

Thus, the computation of $p^q_{max}(\phi_1 \mathcal{U}^{\leq k}\phi_2)$ can be carried out in k iterations, each similar to the process described for Optimal Next (Sec. IV-A.1). The additional step here is that after each maximization operation, the entries of the resultant vector corresponding to states Q^{yes} and Q^{no} are replaced with 1 and 0 respectively. This step is performed to guarantee that the state-indexed vector always carries the correct probabilities.The complexity of this algorithm is k matrix-vector multiplication and k maximization operations. The overall policy is time dependent. that is for each time index k, an action is assigned to each satisfying state.

To illustrate the optimal algorithm for "bounded until", again consider the MDP in Fig. 1 and the PCTL formula $\phi = \mathcal{P}_{max=?}[true \mathcal{U}^{\leq 2}\mathbf{R_3}]$. By inspection, we have $Q^{yes} = \{q_3\}$, $Q^{no} = \emptyset$, and $Q^? = \{q_0, q_1, q_2\}$. By following the method presented above, we compute $\overline{x}^1 = (0 \; 0.44 \; 0 \; 1)^T$ and $\mu^{*^1}(q_1) = a_3$. This means that there exits only one state in $Q^?$ that satisfies the formula in one step or less, q_1 with probability 0.44 and action a_3. Another iteration results in $\overline{x}^2 = (0.44 \; 0.444 \; 0 \; 1)^T$ and $\mu^{*^2}(q_0) = a_1$ and $\mu^{*^2}(q_1) = a_2$. Thus, q_1 satisfies the formula with maximum probability of 0.444 in two steps or less with selection of actions a_2 and a_3 in the first and second time steps, respectively. Moreover, q_0 satisfies the formula with probability 0.44 in two steps with the selection of actions a_1 at q_0 in the first time step and a_3 at q_1 in the second time step.

2) Bounded Until (Stationary) - $\phi = \mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^{\leq k}\phi_2]$: Here, we introduce a sub-optimal algorithm for $\mathcal{U}^{\leq k}$ operator which produces a stationary control policy. This algorithm is used for control synthesis of nested formulas where a stationary policy is required.

The algorithm is essentially the same as the one for Optimal Bounded Until (Sec. IV-B.1) with the exception that the optimal actions determined at each iteration are fixed for the remaining iterations. For instance, consider the example in Sec. IV-B.1 with the formula $\mathcal{P}_{>0.4}[\diamondsuit^{\leq 2}\mathbf{R_3}]$. After the first iteration, we find $\overline{x}^1 = (0 \; 0.44 \; 0 \; 1)^T$ and $\mu(q_1) = a_3$. For the next iteration, we only use action a_3 at q_1 which results in $\overline{x}^2 = (0.44 \; 0.44 \; 0 \; 1)^T$ with policy $\mu(q_0) = a_1$ and $\mu(q_1) = a_3$. Thus, the states q_0 and q_1 satisfy the formula with the stationary policy $\mu(q_0) = a_1$ and $\mu(q_1) = a_3$.

For PCTL formulas of the form $\phi = \mathcal{P}_{\bowtie p}[\phi_1 \mathcal{U}^{\leq k}\phi_2]$, it is theoretically possible to find all the satisfying policies. This becomes important for completeness of the solution for nested formulas (Sec. IV-D). However, it only can be achieved by enumerating every satisfying path, leads to exponential growth in the complexity of the algorithm. Thus, for large MDPs and time index k, finding all the satisfying policies might be impracticable. For this reason, we only use the optimal and stationary algorithms shown above for $\mathcal{U}^{\leq k}$.

C. Until Operator - $\phi = \mathcal{P}_{max=?}[\phi_1 \mathcal{U}\phi_2]$

Here, we are interested in computing probabilities $p^q_{max}(\phi_1 \mathcal{U}\phi_2)$ over all policies and finding the control strategy that gives rise to these optimal probabilities. To solve this problem, again begin by dividing Q into the three subsets Q^{yes} (states satisfying the formula with probability 1), Q^{no} (states satisfying the formula with probability 0), and $Q^?$ (the remaining states).

The computation of optimal probabilities for the states in $Q^?$ is in fact the Maximal Reachability Probability Problem [19]. Thus, we can compute these probabilities by solving the following linear programming problem.

$$\text{Minimize} \sum_{q_i \in Q^?} x_{q_i} \text{ subject to:}$$

$$x_{q_i} \geq \sum_{q_j \in Q^?} \sigma^{q_i}_a(q_j) \,.\, x_{q_j} + \sum_{q_j \in Q^{yes}} \sigma^{q_i}_a(q_j),$$

for all $q_i \in Q^?$ and $(a, \sigma_a) \in Steps(q_i)$.

The problem admits a unique optimal solution, and the actions that give rise to this optimal solution at every state can be identified. Hence, the stationary control policy that produces the maximum probability that satisfies the specification can be obtained. The above linear programming problem can be solved using classical techniques such as the Simplex method, ellipsoid method, or value iteration. The complexity is polynomial in the size of the MDP.

To illustrate this method, again consider the MDP in Fig. 1 with the specification $\mathcal{P}_{max=?}[\neg\mathbf{R}_3\,\mathcal{U}\,\mathbf{R}_2]$. Since state q_2 is the only one satisfying the formula with probability one and q_3 is the only one that fails the formula with probability one, we have the $Q^{yes} = \{q_2\}$, $Q^{no} = \{q_3\}$, and $Q^{?} = \{q_0, q_1\}$. From this we have that $x_{q_2} = 1$ and $x_{q_3} = 0$. The solution to the linear optimization problem can be found to be $x_{q_0} = x_{q_1} = 0.56$ under the policy $\mu(q_0) = a_1$ and $\mu(q_1) = a_3$.

D. Nesting $\mathcal{P}$-operators

Since each probabilistic operator is a state formula itself, it is possible to combine these operators by nesting one inside another. Such a combination of $\mathcal{P}$-operators allows more expressivity in PCTL formulas.

It should be noted that we require all inner $\mathcal{P}$-operators to be of the form $\mathcal{P}_{\bowtie p}[\psi]$ as opposed to being $\mathcal{P}_{max=?}[\psi]$. This is required because each nested probabilistic operator needs to identify a set of satisfying states. Generally, the nested formulas can be written in one of the following forms:

$$\phi = \mathcal{P}_{max=?}[X\phi_R], \quad (2)$$
$$\phi = \mathcal{P}_{max=?}[\phi_L\mathcal{U}^{\leq k}\phi_R], \quad (3)$$
$$\phi = \mathcal{P}_{max=?}[\phi_L\mathcal{U}\phi_R], \quad (4)$$

where ϕ_R in formula (2) and at least one of ϕ_L and ϕ_R in formulas (3) and (4) include a $\mathcal{P}$-operator. Subscripts L and R stand for to the Left and Right of the temporal operator respectively.

Our method of producing a control strategy treats each probabilistic operator individually and proceeds as follows. First, we find the set of initial states Q_{ϕ_R}, from which ϕ_R is satisfied. The corresponding control policy μ_{ϕ_R} is also determined. This is achieved by applying the optimal algorithms shown in Sec. IV-A.1, IV-B.1, and IV-C.

Next, ϕ_L is considered. The set of initial states Q_{ϕ_L} and the corresponding control policy μ_{ϕ_L} are determined. For ϕ_L, it is desired to find all the satisfying stationary policies. This is important for completeness of our solution. The PCTL formulas (3) and (4) require to reach a state in Q_{ϕ_R} only by going through Q_{ϕ_L} states. Thus, at Q_{ϕ_L} states, all and only the actions that satisfy ϕ_L are to be considered. Nevertheless, finding all satisfying policies is only feasible for the temporal operator X (Sec. IV-A.2). For operators $\mathcal{U}^{\leq k}$ and $\mathcal{U}$ in ϕ_L, we use the stationary and optimal algorithms shown in Sec. IV-B.2 and IV-C, respectively, to find μ_{ϕ_L}.

Then, we construct a new MDP $\mathcal{M}' \subseteq \mathcal{M}$ by eliminating the actions that are not allowed by μ_{ϕ_L} from states Q_{ϕ_L}. In other words, we remove all the action choices at states Q_{ϕ_L} except those allowed by μ_{ϕ_L} in $\mathcal{M}$. This step is performed to ensure the satisfaction of the path formula in ϕ_L. If this process results in states with no outgoing transition (blocking states), a self-transition is added to each of these states. This guarantees a new non-blocking MDP. In the last step, the optimal control algorithm is applied for the outer-most $\mathcal{P}$-operator on the modified MDP $\mathcal{M}'$ to find the optimal control policy μ_ϕ and its corresponding probability value p_0 from initial state q_0.

It should be noted that, by the nature of the PCTL formulas, the execution of the optimal policy μ_ϕ only guarantees satisfaction of a formula ϕ which specifies that the system should reach a state in Q_{ϕ_R} through the states in Q_{ϕ_L}. Hence, the path formula specified in ϕ_R is not satisfied by μ_ϕ unless μ_{ϕ_R} is also executed. To ensure the execution of all the specified tasks in ϕ and ϕ_R, we construct a history dependent control policy of the following form:

μ : "Apply policy μ_ϕ until a state in Q_{ϕ_R} is reached. Then, apply policy μ_{ϕ_R}."

For the same reason as state above, the returned probability value p_0 is the maximum probability of satisfying ϕ (reaching a state in Q_R through states of Q_L) under μ_ϕ from initial state q_0. The probability of satisfying the path formula in ϕ_R from q_0 by executing policy μ cannot be found directly because it is not known which state in Q_R is reached first. However, since the probability of satisfying ϕ_R from each state in Q_R is available, a bound on the probability of satisfying ϕ and then ϕ_R from q_0 can be defined. The lower and upper limits of this bound are $p_0 p^{min}_{\phi_R}$ and $p_0 p^{max}_{\phi_R}$, respectively, where $p^{min}_{\phi_R}$ and $p^{max}_{\phi_R}$ denote the minimum and maximum probabilities of satisfying ϕ_R from Q_{ϕ_R} respectively.

To illustrate the control synthesis algorithm of nested formulas, consider again the MDP shown in Fig. 1 with the formula $\mathcal{P}_{max=?}[\mathcal{P}_{\leq 0.50}[X\,\mathbf{R_2}]\,\mathcal{U}^{\leq 2}\,\mathbf{R_3}]$. Since there is no $\mathcal{P}$-operator on the right side of $\mathcal{U}^{\leq 2}$, the algorithm proceeds with finding the all initial states and actions for $\phi_L = \mathcal{P}_{\leq 0.55}[X\,\mathbf{R_2}]$. By applying the All Next algorithm (Sec. IV-A.2), we find the satisfying actions $\{a_1\}$ at q_0, $\{a_2, a_4\}$ at q_1, $\{a_4\}$ at q_2, and $\{a_1, a_4\}$ at q_3. Next, a new MDP $\mathcal{M}'$ is constructed by eliminating action a_3 at q_1 and a_1 at q_2 which do not satisfy ϕ_L. By performing the optimal control algorithm for "bounded until" (Sec. IV-B.1) on $\mathcal{M}'$, we find the maximum probability of satisfying ϕ to be $\bar{x}^1 = (0\ 0.4\ 0\ 1)^T$ with the policy $\mu^1(q_1) = a_2$ and $\bar{x}^2 = (0.4\ 0.44\ 0\ 1)^T$ with $\mu^2(q_0) = a_1$ and $\mu^2(q_1) = a_2$. Thus, the maximum probability of satisfying ϕ from initial states q_0 is 0.4 with the policy $\mu(q_0) = a_1$ and $\mu(q_1) = a_2$.

E. Correctness, Completeness, and Complexity

Our solution to Problem 1 is correct by construction but conservative for nested formulas. As mentioned above, for completeness of the solution, we need to consider all the actions that satisfy ϕ_L at each state of Q_{ϕ_L}. However, due to computational complexity, we use the stationary and optimal algorithms for $\mathcal{U}^{\leq k}$ and $\mathcal{U}$ operators, respectively, which return only the optimal actions as opposed to all satisfying actions. Hence, our solution for the formulas whose ϕ_L

include "bounded until" or "until" operators is not complete. In fact, for these formulas, the algorithm may return a suboptimal policy or may not find a solution at all even though one might exist.

Nevertheless, our solution for the group of nested PCTL formulas where ϕ_L does not include $\mathcal{U}^{\leq k}$ or $\mathcal{U}$ is complete. This group of specifications is useful in robotic applications. In these applications, tasks such as "Eventually reach A and then reach B while always avoiding C" or "Eventually reach A through regions from which the probability of convergence to C is less than 0.30" are of interest.

The overall time complexity for PCTL control synthesis for an MDP from a formula ϕ is linear in the size of the formula and polynomial in the size of the model. That is because each operator is treated separately. Moreover, the most expensive case is the "until" operator, for which we must solve a linear optimization problem of size of the model, $|\mathcal{M}|$. Using, for example, the ellipsoid method, this can be done in polynomial time. For MDPs, we define the size of the model to be $\sum_{q_i \in Q} |\mathcal{A}(q_i)|$, the total number of actions available at each state.

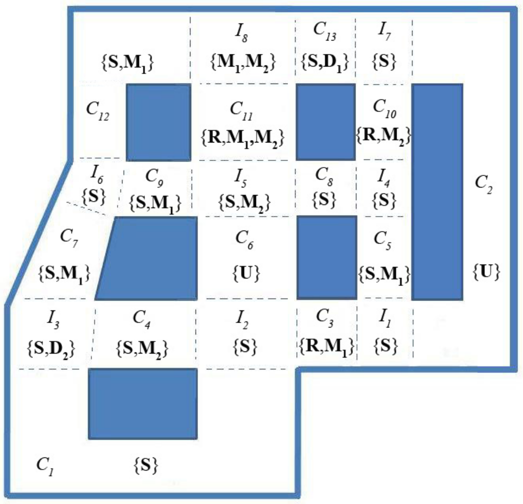

Fig. 3. Schematic representation of an indoor environment. Each region has a unique identifier ($C_1, \ldots, C_{13}$ for corridors and $I_1, \ldots, I_8$ for intersections, respectively). The properties satisfied at the regions are shown between curly brackets inside the regions: **S** = *Safe*, **R** = *Relatively safe*, **U** = *Unsafe*, $\mathbf{M_1}$ = *Medical supply type 1*, $\mathbf{M_2}$ = *Medical supply type 2*, $\mathbf{D_1}$ = *Destination 1*, and $\mathbf{D_2}$ = *Destination 2*.

V. Case study

In this section, we apply the method described above to the provably-correct deployment of a mobile robot with noisy sensors and actuators using a simulator.

For this problem, we considered the environment whose topology is schematically shown in Fig. 3. It consists of corridors of various widths and lengths ($C_1, \ldots, C_{13}$) and intersections of several shapes and sizes ($I_1, \ldots, I_8$). There are six properties of interest about the regions: *Safe* (the robot can safely drive through a corridor or intersection with this property), *Relatively safe* (the robot can pass through the region but should avoid it if possible), *Unsafe* (the corresponding region should be avoided), *Medical supply 1* and *2* (there are medical supplies of type 1 and 2 in the regions associated with these properties respectively), and *Destinations 1* and *2* (regions to be visited).

For deployment of the robot, we first considered a set of feedback control primitives for the robot. Due to the presence of input-output noise, the outcome of each control primitive is characterized probabilistically. Assuming that the robot can determine exactly its current region in the environment, then the primitives and the transition probabilities yield an MDP model of the motion of the robot in the environment (see [17] for a detailed discussion on creating such models).

With this MDP and a PCTL formula expressed over properties of the regions of the environment, we used the framework described in this paper to determine a control strategy to satisfy the formula. Note that a control strategy is an assignment of a control primitive to each region of the environment that the robot visits given the history of the visited regions.

A. Simulation Tool

For the construction of the MDP and deployment of the mobile robot, we used our RIDE Simulator (Fig. 4). As shown in [17], the simulator mimics the motion of an iRobot iCreate platform with a Hokoyu URG-04LX laser range finder, and APSX RW-210 RFID reader in an indoor environment. Specifically, it emulates experimentally measured response times, sensing and control errors, and noise levels and distributions in the laser scanner readings.

The robot is able to execute one of the following four controllers (actions) - FollowRoad, GoRight, GoLeft, and GoStraight. These controllers utilize data from the laser scanner to steer the robot according to the descriptive title. Due to noise in the actuators and sensors, however, the resulting motion may be different than intended. Details can be found in [17].

B. Construction of the MDP model

Each state of the MDP is a collection of regions such that the Markovian property is satisfied (*i.e.,* the result of an action at a state depends only on the current state). The set of actions available at a state is the set of controllers available at the last region in the set of regions corresponding to the state. More details on the construction of the MDP are given below.

The environment (Fig. 3) consists of 13 corridors, within which only the controller FollowRoad is available. There are also two 4-way and six 3-way intersections in the environment. The controllers available at 4-way intersections are GoRight, GoLeft, and GoStraight, while at the 3-way intersections only GoRight and GoLeft controllers are available. Through extensive trials, we concluded that, by grouping two adjacent regions (a corridor and an intersection) in a state, we achieve the Markovian property, for all pairs of adjacent regions. For example, the connecting regions of C_1-I_2 represent one state of the MDP, which has transitions to states I_2-C_3, I_2-C_4, and I_2-C_6 enabled by actions GoRight, GoLeft,

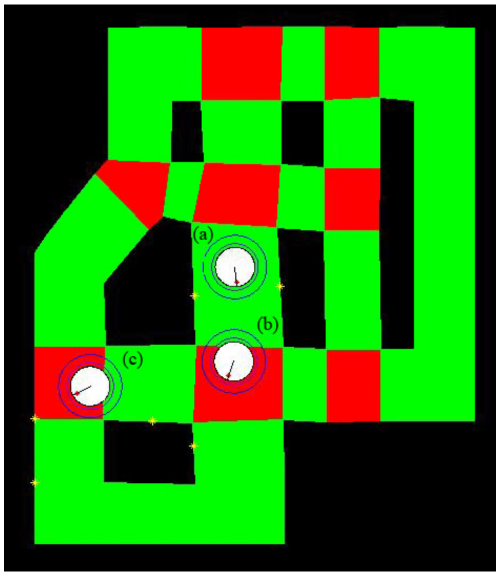

Fig. 4. Snapshots from the RIDE simulator. The robot is represented as a white disk. The arrow inside the white disk shows the robot's heading. The inner circle around the robot represents the "emergency" radius (if there is an obstacle within this zone, the emergency controller is used). The outer circle represents the radius within which the forward speed of the robot varies with the distance to the nearest obstacle. If there are no obstacles in this area, the robot moves at maximum speed. The yellow dots are the laser readings used to define the target angle. (a) The robot centers itself on a stretch of corridor by using FollowRoad; (b) The robot applies GoRight in an intersection; (c) The robot applies GoLeft.

and GoStraight. When designing the robot controllers, we also made sure that the robot never gets stuck in a region, *i.e.,* the robot can only spend a finite amount of time in each region. Thus, the states are of the form intersection-corridor and corridor-intersection (states such as I_i-I_i or C_i-C_i do not exist). The resulting MDP for the environment shown in Fig. 3 has 52 states. The set of actions available at a state of the MDP is the set of controllers available at the second region of the state. For example, when in state C_1-I_2 only those actions from region I_2 are allowed.

To obtain transition probabilities, we performed a total of 500 simulations for each controller available in each MDP state. In each trial, the robot was initialized at the beginning of the first region of each state. If this region was a corridor, then the FollowRoad controller was applied until the system transitioned to the second region of the state. If the first region was an intersection then the controller most likely to transition the robot to the second region was applied. Once the second region was reached, one of the allowed actions was applied and the resulting transition was recorded. The results were then compiled into the transition probabilities.

The set of properties of the MDP was defined to be $\Pi = \{\mathbf{S}, \mathbf{R}, \mathbf{U}, \mathbf{M_1}, \mathbf{M_2}, \mathbf{D_1}, \mathbf{D_2}\}$, where $\mathbf{S}$ = *Safe*, $\mathbf{R}$ = *Relatively safe*, $\mathbf{U}$ = *Unsafe*, $\mathbf{M_1}$ = *Medical supply type 1*, $\mathbf{M_2}$ = *Medical supply type 2*, $\mathbf{D_1}$ = *Destination 1*, and $\mathbf{D_2}$ = *Destination 2*. Each state of the MDP was mapped to the set of properties that were satisfied at the second region of the state (Fig. 3).

C. Case Studies

Consider the RIDE configuration and the following three motion specifications:

Specification 1: "Reach *Destination 1* by always avoiding *Unsafe* regions."

Specification 2: "Reach *Destination 1* by going through the regions from which the probability of converging to a *Relatively safe* region is less than 0.50 and always avoiding *Unsafe* regions."

Specification 3: "Reach *Destination 1* by avoiding *Unsafe* and *Relatively safe* regions if *Medical supply 1* is not available at such regions, and then reach *Destination 2* by driving through regions that are *Safe* or at which *Medical supply 2* is available with probability greater than or equal to 0.50."

Given that we are interested in the policy that produces the maximum probability of satisfying each specification, Specifications 1, 2, and 3 translate naturally to the PCTL formulas ϕ_1, ϕ_2, and ϕ_3, respectively, where

$$\phi_1 \;:\; \mathcal{P}_{max=?}\,[\neg\mathbf{U}\,\mathcal{U}\,\mathbf{D_1}\,] \tag{5}$$
$$\phi_2 \;:\; \mathcal{P}_{max=?}\,[(\mathcal{P}_{<0.50}[X\,\mathbf{R}] \wedge \neg\mathbf{U})\,\mathcal{U}\,\mathbf{D_1}\,] \tag{6}$$
$$\phi_3 \;:\; \mathcal{P}_{max=?}\,[\neg\mathbf{U} \wedge \neg(\mathbf{R} \wedge \neg\mathbf{M_1})\,\mathcal{U}\,(\mathbf{D_1} \wedge \mathcal{P}_{\geq 0.50}[(\mathbf{S} \vee \mathbf{M_2})\,\mathcal{U}\,\mathbf{D_2}])\,] \tag{7}$$

Assuming that the robot is initially in C_1-I_2 (so that physically it is in C_1 and oriented towards I_2), we used the computational framework described in this paper to find control strategies maximizing the probabilities of satisfying the above specifications. The maximum probabilities for Specifications 1 and 2 were 0.862 and 0.152 respectively. The maximum probability of Specification 3 was 0.451, and the probability of reaching the second destination determined in this specification was 0.269. It should be noted that we were able to produce an exact number instead of a bound for the probability of satisfaction of the nested formula (7) because even though two states satisfy $\mathbf{D_1}$ (I_8-C_{13} and I_7-C_{13}) only one (I_8-C_{13}) is reachable from the initial state. To confirm these predicted probabilities, we performed 500 simulation for each of the control strategies. The simulations showed that the probabilities of satisfying ϕ_1 and ϕ_2 were 0.838 and 0.118 respectively. The rate of successful runs for the strategy obtained from ϕ_3 was 0.435 to reach *destination 1* and 0.244 to reach *destination 1* and then *2*. The small discrepancy between the theoretical values and simulation results is likely due to remaining non-Markovian behavior of the transitions.

Snapshots from a movie showing a motion of the robot produced by the control strategy maximizing the probability of satisfying Specification 3 is shown in Fig. 5. With reference to the notation defined in Fig. 3, it can be seen that the robot follows the route $C_1I_2C_3I_1C_5I_4C_8I_5C_9I_6C_{12}I_8C_{13}I_7C_{10}I_4C_8I_5C_9I_6C_7I_3$. It can be easily seen that this run satisfies Specification 3 (ϕ_3 in (7)) because $\neg\mathbf{U} \wedge \neg(\mathbf{R} \wedge \neg\mathbf{M_1})$ is true until $\mathbf{D_1}$ is satisfied and then $(\mathbf{S} \vee \mathbf{M_2})$ is true before $\mathbf{D_2}$ becomes

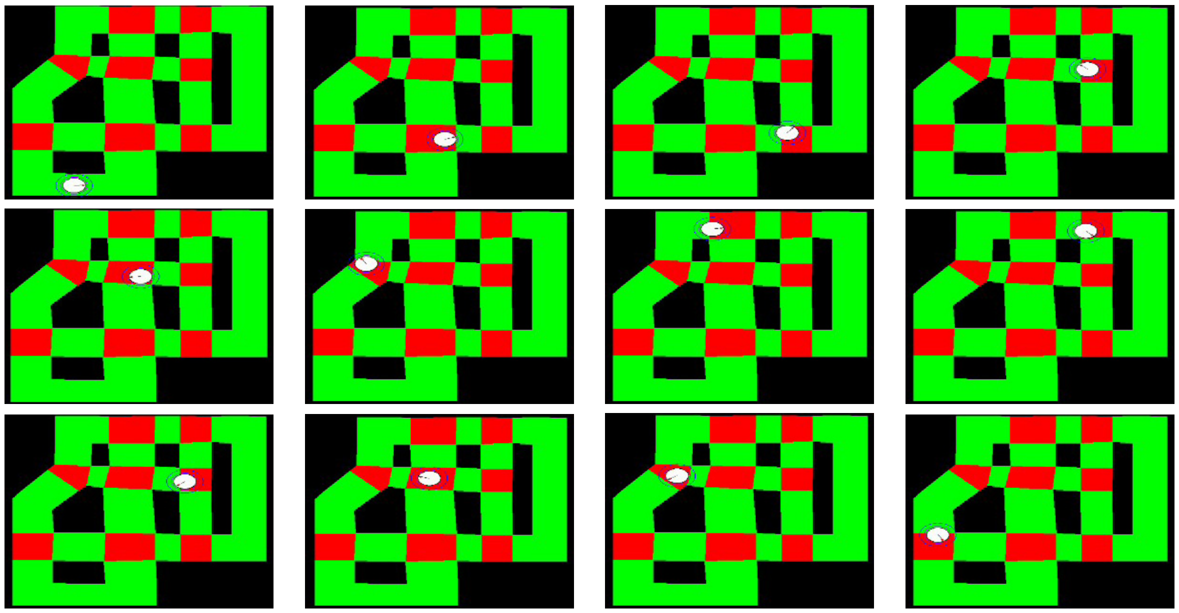

Fig. 5. Snapshots (to be read left-to-right and top-to-bottom) from a movie showing a robot motion produced by applying the control strategy maximizing the probability of satisfying Specification 3 (ϕ_3 in (7)).

true. This movie, together with other movies of simulation trials obtained by applying the above control strategies are available for download from [18].

VI. Conclusion

We presented a computational framework for control synthesis for an MDP that maximizes the probability of satisfying a Probabilistic Computation Tree Logic (PCTL) formula. The approach provides a rich specification language and probabilistic guarantees on the system performance. The method was demonstrated through simulations of a mobile robot being deployed from PCTL specifications in a partitioned environment.

Acknowledgements

The authors would like to thank K. Ryan and B. Chang-Yun Hsu from Boston University for their help with the development of the RIDE simulator.

References

[1] C. Baier, "On algorithmic verification methods for probabilistic systems," Ph.D. dissertation, University of Mannheim, Germany, 1998.

[2] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing*, vol. 6, pp. 102–111, 1994.

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

[4] J. Barnat, L. Brim, I. Cern, M. Ceka, and J. Tumov, "Distributed qualitative ltl model checking of markov decision processes," in *In Proceedings of 5th International Workshop on Parallel and Distributed Methods in verifiCation*, Bonn, Germany, 2006, pp. 1–15.

[5] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis, "Multi-objective model checking of Markov decision processes," *Logical Methods in Computer Science*, vol. 4, no. 4, pp. 1–21, 2008.

[6] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker, *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), ser. CRM Monograph Series. American Mathematical Society, 2004, vol. 23.

[7] C. Baier and M. Kwiatkowska, "Model checking for a probabilistic branching time logic with fairness," *Distributed Computing*, vol. 11, pp. 125–155, 1998.

[8] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *Proceedings of the IEEE Conference on Decision and Control*, 2004.

[9] M. M. Quottrup, T. Bak, and R. Izadi-Zamanabadi, "Multi-robot motion planning: A timed automata approach," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 2004, pp. 4417–4422.

[10] H. K. Gazit, G. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *IEEE Conference on Robotics and Automation*, Rome, Italy, 2007.

[11] G. E. Fainekos, S. G. Loizou, and G. J. Pappas, "Translating temporal logic to controller specifications," in *Proceedings of the IEEE Conference on Decision and Control*, 2006.

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

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

[14] E. A. Emerson, *Handbook of Theoretical Computer Science: Formal Models and Semantics*. North-Holland Publishing Company and MIT Press, 1990, vol. B, ch. Temporal and Modal Logic, pp. 995–1072.

[15] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," in *VMCAI*, Charleston, SC, 2006, pp. 364–380.

[16] M.Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Hybrid Systems: Computation and Control: 11th International Workshop*, ser. Lecture Notes in Computer Science, M. Egerstedt and B. Mishra, Eds. Springer Berlin / Heidelberg, 2008, pp. 287–300.

[17] M. Lahijanian, J. Wasniewski, S. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees,," in *IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, 2010.

[18] "Robotic indoor environment (ride)." [Online]. Available: hyness.bu.edu/ride/

[19] L. de Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanfod University, 1994.