Path Planning Using Formal Methods

Apurva Patil

1 Introduction

Motion planning is a fundamental problem that has received a lot of attention from robotics community. The classical goal of motion planning is to generate a feasible path for a robot to move from an initial configuration to a target configuration while avoiding obstacles on the way. Various methods such as potential fields, navigation functions, cell decomposition, and sampling based have been employed in the literature. Formal synthesis is another approach used for this purpose. The attractive point of formal method is its ability to express complex specifications using Boolean, temporal operators and atomic propositions. For example, specifications like "Visit A, B, C infinitely often and always avoid D" can be easily expressed in formal methods.

In this report, we solve the motion planning problem for continuous space using formal methods. We consider two types of settings: one in which there are no uncertainties in the system model and the environment, and other in which uncertainties are taken into account. There are model checking algorithms available in the literature that can be used to generate motion plans and control policies for a finite model of robot motion. Various abstraction techniques are proposed in the past to obtain a finite state model from the continuous space and/or time model. In this report, we consider two different types of abstraction techniques, one is applied in the setting when there are no uncertainties, which results into a finite transition system, and the other one is used for the setting with uncertainties, which results into a Markov Decision Process (MDP). The algorithm for the no uncertainties case is based on a sampling based path-planning algorithm. In this algorithm, as the number of samples increases, the probability that a satisfying path is found approaches 1, i.e. our algorithm is probabilistically complete.

Throughout this report, we have used LTL formulae for expressing the specifications formally. LTL has very friendly syntax and semantics, which can be easily translated into natural language and can be used by untrained human operators. Also, we make an assumption that during the motion of the robot in the environment, the robot can determine its position precisely.

2 Preliminaries

Given a set Z, let 2^Z denote its power set and let $|Z|$ denote its cardinality. We give the following formal definitions.

Definition 2.1 (Finite Transition System): A finite transition system is a tuple $\mathcal{T} = (X, x_0, \Delta, AP, J)$, where:

- X is a finite set of states;
- $x_0 \in X$ is the initial state;
- $\Delta \subseteq X \times X$ is a set of transitions;
- AP is a set of atomic propositions (properties of interest);
- $J : X \to 2^{AP}$ is a labeling function.

Definition 2.2 (Markov Decision Process): An MDP is a tuple $\mathcal{M} = (Q, q_0, \mathcal{A}, \mathbb{P}, AP, K)$ where:

- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- $\mathcal{A}$ is a set of available actions, with $\mathcal{A}(q)$ defining the set of actions available at the state q;
- $\mathbb{P} : Q \times \mathcal{A} \times Q \to [0, 1]$ is a transition probability function such that $\forall$ $q \in Q$ and $\forall$ $\alpha \in \mathcal{A}(q)$, $\sum_{q' \in Q} \mathbb{P}(q, \alpha, q') = 1$;
- AP is a set of atomic propositions;
- $K : Q \to 2^{AP}$ is a labeling function.

3 Problem Formulation

Let $\mathcal{D} \subseteq \mathbb{R}^n$ be the configuration space of a robot, where $n \in \mathbb{N}$, $n \geq 2$. Let $\mathcal{R}$ be a set of disjoint regions in $\mathcal{D}$. Let AP be a set of atomic propositions, i.e. the properties of interest corresponding to the regions in $\mathcal{R}$. Throughout this report, we consider $AP = \{\text{obstacle}, \text{goal}_i, \text{free}\}$, where $i = \{1, 2, \ldots, s\}$ and s is the total number of goal regions in $\mathcal{D}$. A map L: $\mathcal{R} \to 2^{AP}$ specifies how properties are associated to the region in $\mathcal{R}$.

Problem 3.1: Given an environment, $\mathcal{E} = (\mathcal{D}, x_0, \mathcal{R}, AP, L)$, the initial configuration of the robot $x_0 \in \mathcal{D}$ and an LTL formula ϕ over the set of properties AP, find a control policy that generates a satisfying trajectory (in the absence of uncertainties) or that maximizes the probability of satisfying ϕ (in the presence of uncertainties).

In the Section 4, we solve Problem 3.1 assuming there are no uncertainties in the environment and the model. To solve the problem in this setting we use a sampling based algorithm Probabilistic RoadMap (PRM). A transition system $\mathcal{T}$ is constructed using a PRM-based algorithm and then formal synthesis is used to generate a trajectory of $\mathcal{T}$ that satisfies ϕ. As PRM possesses the theoretical guarantee of probabilistic completeness, the proposed mechanism should find

a satisfying trajectory with probability 1 if such a trajectory exists and the number of samples approaches infinity.

In the Section 5, we solve Problem 3.1 in the presence of uncertainties. We abstract the environment $\mathcal{E} = (\mathcal{D}, x_0, \mathcal{R}, AP, L)$ into an MDP and use probabilistic synthesis techniques to generate a policy that maximizes the probability of satisfying ϕ.

4 Planning in the Absence of Uncertainties

The mechanism implemented in this setting consists of three stages. First, in Section 4.1, we construct a graph $G = (V, E)$ using a PRM-based algorithm, where $V \subset \mathcal{D}$ is a set of vertices and $E \subseteq V \times V$ is a set of edges in the graph G. Next, in Section 4.2, we construct a finite transition system $\mathcal{T}$ using the graph G. This way we abstract the continuous environment $\mathcal{E}$ into a finite transition system $\mathcal{T}$. Lastly, in Section 4.3, we synthesize a trajectory of $\mathcal{T}$ that satisfies the LTL formula ϕ.

4.1 Construction of a Graph G

The basic idea of PRM is the construction of a graph, G by attempting connections among m random samples in $\mathcal{D}$. We modify the algorithm of PRM so that it can be used to construct a finite transition system $\mathcal{T}$ for the formal synthesis purpose. We first briefly introduce the functions used by our modified-PRM algorithm.

4.1.1 Sample

Let *Sample*: $\omega \to \{Sample_i(\omega)\}_{i \in \mathbb{N}} \subset \mathcal{D}$ be a map from a sample space Ω to a sequence of independent and identically distributed (i.i.d.) samples in $\mathcal{D}$, from a given distribution P. We assume that the support of P is the entire configuration space $\mathcal{D}$.

4.1.2 Near

Given a graph G, a configuration $d \in \mathcal{D}$, and a positive real number $r \in \mathbb{R}_{>0}$, the function *Near*: $(G, d, r) \to V' \subseteq V$ returns the vertices in V that are contained in a ball of radius r centered at d, i.e.

$$Near(G = (V, E), d, r) \coloneqq \{v \in V : v \in \mathcal{B}_{d,r}\}$$

4.1.3 isSimpleEdge

isSimpleEdge: $\mathcal{D} \times \mathcal{D} \to \{0, 1\}$ is a function that takes two configurations d_1, d_2 in $\mathcal{D}$ and returns 1 if the edge (d_1, d_2) ($\{d \in \mathbb{R}^n : d = \lambda d_1 + (1-\lambda) d_2, \lambda \in [0, 1]\}$) is simple, otherwise it returns 0. An edge (d_1, d_2) is simple if $(d_1, d_2) \subset \mathcal{D}$ and the number of times (d_1, d_2) crosses the boundary of any region $R \in \mathcal{R}$ is at

most one. Therefore, *isSimpleEdge* returns 1 if either:
(1) d_1 and d_2 belong to the same region R and (d_1, d_2) does not cross the boundary of R or
(2) d_1 and d_2 belong to two regions R_1 and R_2, respectively, and (d_1, d_2) crosses the common boundary of R_1 and R_2 once. In Algorithm 1, an edge is constructed between d_1 and d_2 if it is a simple edge (i.e *isSimpleEdge* returns 0).

Our algorithm of constructing G using modified-PRM is outlined in Algorithm 1. The algorithm begins with initializing the vertex set V with the initial configuration of the robot x_0, and m randomly selected i.i.d. configurations from $\mathcal{D}$. The edge set E is initialized with an empty set. Then the algorithm attempts to connect vertices within a distance of r. Successful connections (that correspond to simple edges) result in the addition of new edges to the edge set E. Note that if the connection between vertices v and u is successful then two edges (v, u) and (u, v) are added to the set E. The construction of G is explained in Fig. 1.

Algorithm 1: Modified-PRM

```
1: V ← {x_0} ∪ {Sample_i}_{i=1,...,m}; E ← ∅
2: foreach v ∈ V do
       U ← Near(G = (V, E), v, r)\{v};
       foreach u ∈ U do
           if isSimpleEdge(v, u) then
               E ← E ∪ {(v, u), (u, v)}
3: return G = (V, E)
```

4.2 Construction of a Transition System $\mathcal{T}$

Once the the graph $G = (V, E)$ is constructed the transition system $\mathcal{T} = (X, x_0, \Delta, AP, h)$ is constructed as follows:

- $X = V$;
- $\Delta = E$;
- $\forall\, x \in X$, $J(x) = L(R_k)$, where $R_k \in \mathcal{R}$ is a region in $\mathcal{D}$ such that $x \in R_k$.

Due to the function *isSimpleEdge* used in the construction of G, the satisfaction of the LTL formula ϕ can be checked by only looking at the properties corresponding to the states of the transition system. Note that, all the edges in the set E are bidirectional; hence if there is a transition from state x_i to x_j then there is also a transition from state x_j to x_i. Also note that the vertex set V of the graph G includes the initial configuration of the robot x_0.

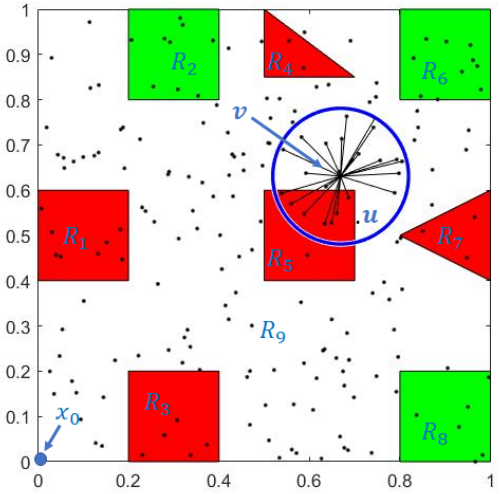


Figure 1: The configuration space is $\mathcal{D} \subseteq \mathbb{R}^2$. The initial configuration $x_0 = (0, 0)$. A set of disjoint regions $\mathcal{R} = \{R_i\}_{i=1,\ldots,9}$. $L(R_k) = \{obstacle\}$ for $k = 1, 3, 4, 5, 7$, $L(R_2) = \{goal_1\}$, $L(R_6) = \{goal_2\}$, $L(R_8) = \{goal_3\}$ and $L(R_9) = \{free\}$. The black dots represents random i.i.d. samples in $\mathcal{D}$. A ball $\mathcal{B}_{d,r}$ of radius $r = 0.2$ centered at $v \in R_9$ is shown in blue. The connections of v with the configurations inside the ball $\mathcal{B}_{d,r}$ are attempted. Only the connections which correspond to the simple edges are included in the edge set E. The edges are shown by black lines. The configuration u is inside the ball $\mathcal{B}_{d,r}$ but the edge (v, u) crosses the boundary of R_5 twice; hence (v, u) is not included in the edge set E.

4.3 Synthesis of a Satisfying Trajectory

At this stage, we can use the model checking tools to find a trajectory π such that $\pi \models \phi$, using closed system synthesis. The transition system $\mathcal{T}$ and the negation of the LTL formula $\neg\phi$ are fed to the model checking tool. If there exists a trajectory $\pi \models \phi$ the model checker spits a counterexample, which is a satisfying trajectory.

Note that the transition system $\mathcal{T}$ is an under-approximation of the environment $\mathcal{E} = (\mathcal{D}, x_0, \mathcal{R}, AP, L)$. Hence, the trajectory synthesized in $\mathcal{T}$ can be implemented in $\mathcal{E}$. Also note that the modified-PRM algorithm retains the probabilistic completeness of PRM (for the proof of probabilistic completeness of PRM please refer to (1)). Hence, as the number of samples in the construction of the graph G approaches infinity, the proposed mechanism finds a trajectory that satisfies the LTL formula ϕ with probability 1 if such a trajectory exists.

4.4 Example

Consider the environment $\mathcal{E}$ shown in Fig. 1. $\mathcal{D} = (0,1)^2$. The LTL formula $\phi = \square((\neg obstacle) \wedge (\lozenge goal_1) \wedge (\lozenge goal_2) \wedge (\lozenge goal_3))$, i.e. we want to find a trajectory that starts from x_0 and visits R_2, R_6 and R_8 infinitely often while avoiding obstacles on the way. The first two stages of the presented mechanism (Sections 4.1 and 4.2) are implemented in MATLAB. The construction of a graph using the modified-PRM algorithm is shown in Fig. 2 (a)-(e). The number of vertices in the graph are 100. Hence, $|X| = 100$. The final stage of the mechanism (Section 4.3) is implemented in NuSMV. The solution is shown in Fig. 2 (f)

5 Planning in the Presence of Uncertainties

The mechanism implemented in this setting consists of two stages. First, we abstract the continuous environment $\mathcal{E}$ into an MDP $\mathcal{M}$ (Section 5.1). Then, we synthesize a control policy $\mathcal{C}^*$ for $\mathcal{M}$ that maximizes the probability of satisfying ϕ using probabilistic synthesis tools, thereby solving Problem 3.1 (Section 5.2).

5.1 Construction of an MDP $\mathcal{M}$

We discretize D into a grid as shown in Fig. 3. Each cell of the grid represents a state $q \in Q$ of $\mathcal{M}$. The initial state of $\mathcal{M}$, q_0 is such that the initial position of the robot $x_0 \in q_0$. The sets of available actions at each state $q \in Q$, and the transition probability function $\mathbb{P}$ will be designed according to the robot's model and the discretization of $\mathcal{D}$. The labeling function K is as follows:

- $\forall\ q \in Q$, if $q \cap R_k \neq \emptyset$ and $L(R_k) = \{obstacle\}$ then $K(q) = L(R_k)$;
- $\forall\ q \in Q$, if $q \subseteq R_k$ and $L(R_k) = \{free\}$ or $\{goal_i\}$ then $K(q) = L(R_k)$.

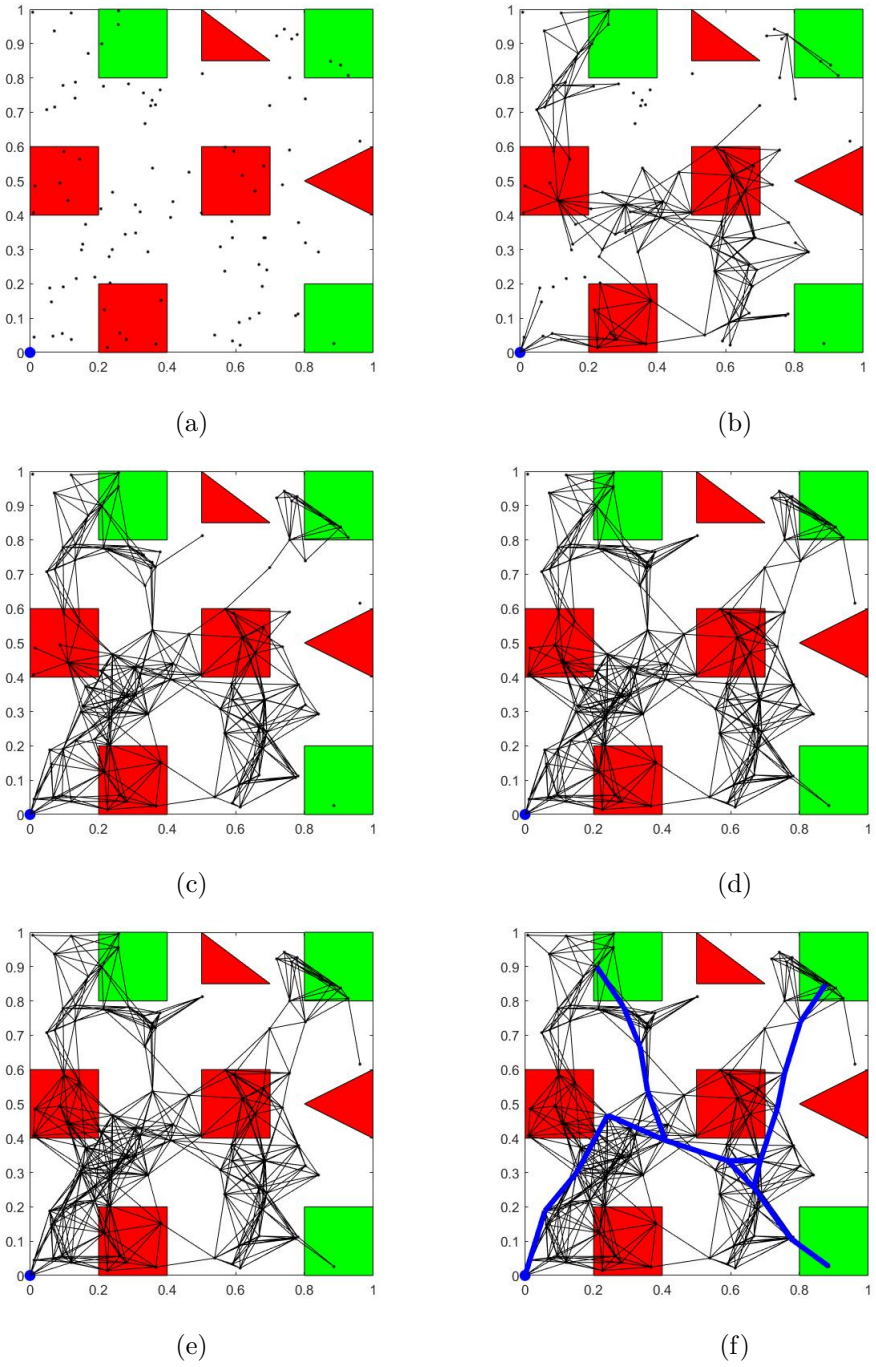


Figure 2: (a)-(e) Construction of a graph G using modified-PRM algorithm. (f) The specification is to visit all the green regions infinitely often while avoiding red regions. A satisfying trajectory obtained using formal synthesis is shown in blue.

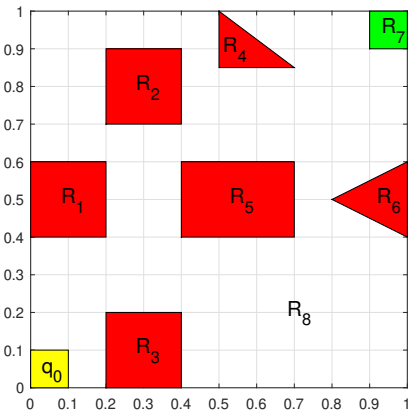


Figure 3: The configuration space is $\mathcal{D} \subseteq \mathbb{R}^2$. The initial state q_0 is shown in yellow. A set of disjoint regions $\mathcal{R} = \{R_i\}_{i=1,\ldots,8}$. $L(R_k) = \{obstacle\}$ for $k = 1, \ldots, 6$, $L(R_7) = \{goal_1\}$, and $L(R_8) = \{free\}$.

5.2 Synthesis of the Optimal Control Policy $\mathcal{C}^*$

Now, we want to generate an MDP control policy $\mathcal{C}^*$ that maximizes the probability of satisfying ϕ, i.e. $\mathcal{P}_{max}[\phi]$. The probabilistic model checking algorithms take ϕ and the MDP $\mathcal{M}$, and return the maximum probability over all possible policies that ϕ is satisfied and a control policy $\mathcal{C}^*$ that produces this probability. The method is as follows. The state space Q is partitioned into three subsets. The first, Q^{yes}, contains all those states that satisfy ϕ with probability 1 for some control policy. The second, Q^{no} contains those states for which the probability of satisfying ϕ is 0 for all control policies, while $Q^?$ contains the remaining states.

Let y_q denote the probability of satisfying ϕ from state $q \in Q$. Then y_q's are determined by the following linear optimization problem:

$$\begin{aligned} \min_{y_q} \sum_{q \in Q} y_q \text{ subject to:} & \\ y_q = 1 \quad & \forall q \in Q^{yes} \\ y_q = 0 \quad & \forall q \in Q^{no} \\ y_q \geq \sum_{q' \in Q} \mathbb{P}(q, \alpha, q') \cdot y_{q'} \quad & \forall \alpha \in \mathcal{A}(q) \\ & \forall q \in Q \backslash (Q^{yes} \cup Q^{no}) \end{aligned} \tag{1}$$

Finding the unique solution to this problem yields the optimal probabilities y_q^*. The desired control policy $\mathcal{C}^*$ is thus the function mapping each state $q \in Q \backslash Q^{no}$ to the optimal action $\alpha_q^* \in \mathcal{A}(q)$ identified by the solution to eq. (1).

5.3 Example

Consider the environment $\mathcal{E}$ shown in Fig. 3. $\mathcal{D} = (0,1)^2$. The LTL formula is $\phi = \neg obstacle\ \mathcal{U}\ goal_1$, i.e. we want to synthesize a control policy $\mathcal{C}^*$ starting from q_0 that maximizes the probability of reaching R_7 while avoiding obstacles on the way. We discretize $\mathcal{D}$ into a grid in which the size of each cell is 0.1×0.1. Hence, $|Q| = 100$. The initial state q_0 is shown in yellow. Assume that the set of available actions for all states $q \in Q$ is $\mathcal{A}(q) = \{right, up, left, down\}$ with the uncertainty of 0.8 of successful transition and 0.1 of moving ± 45 deg to the intended direction. The robot bounces back to its original state when it hits the boundary of $\mathcal{D}$. The transition probabilities can be computed given the sensor, plant, and environment noise model or through experimental trials. The first stage (Section 5.1) of the presented mechanism is implemented in MATLAB. The second stage (Section 5.2) is implemented in the probabilistic verification and synthesis tool PRISM. The maximum probability of satisfying ϕ starting from q_0 is 0.79. The optimal policy $\mathcal{C}^*$ is shown in Fig. 4. One of the trajectories generated using $\mathcal{C}^*$ that satisfies ϕ and another one that doesn't satisfy ϕ are shown in Fig. 5.

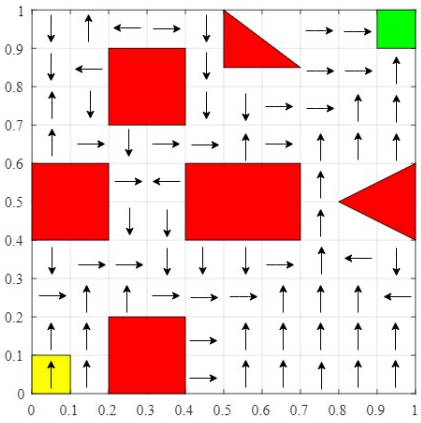
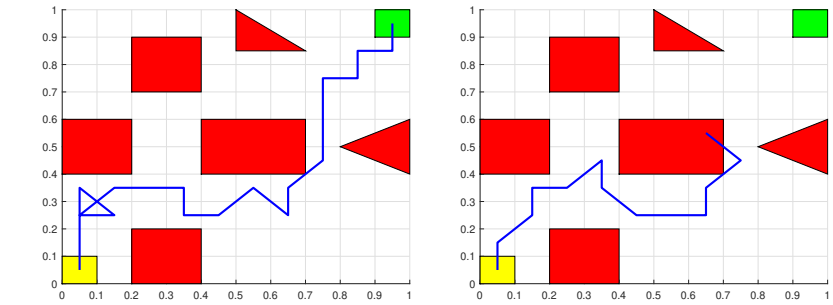


Figure 4: The optimal control policy $\mathcal{C}^*$

6 Conclusion

In this report, we analyzed the motion planning problem for continuous space using formal methods. We considered two different types of abstraction techniques, one is applied in the setting when there are no uncertainties in the system model and the environment, and the other one is used for the setting with uncertainties. The algorithm for the no uncertainties case is based on a sampling based path-planning algorithm PRM. In this algorithm, as the number



(a) (b)

Figure 5: (a) A trajectory (shown in blue) obtained by $\mathcal{C}^*$ that satisfies ϕ. (b) A trajectory obtained by $\mathcal{C}^*$ that doesn't satisfy ϕ.

of samples increases, the probability that a satisfying path is found approaches 1, i.e. our algorithm is probabilistically complete. For the case with uncertainties, we abstract the continuous space into a grid and design an MDP. The probabilistic synthesis tool is used to generate an optimal policy in this MDP.

References

[1] S. Karaman, and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.

[2] C. Vasile and C. Belta, "Sampling-based temporal logic path planning," in Int. Conf. on Intelligent Robots and Systems. IEEE, Nov 2013, pp. 4817–4822.

[3] Yoo C, Fitch R and Sukkarieh S (2013) Provably-correct stochastic motion planning with safety constraints. In: Proceedings of IEEE ICRA, pp. 981–986.

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