2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Negotiating the Probabilistic Satisfaction of Temporal Logic Motion Specifications

Igor Cizelj and Calin Belta

Abstract— **We propose a human-supervised control synthesis method for a stochastic Dubins vehicle such that the probability of satisfying a specification given as a formula in a fragment of Probabilistic Computational Tree Logic (PCTL) over a set of environmental properties is maximized. Under some mild assumptions, we construct a finite approximation for the motion of the vehicle in the form of a tree-structured Markov Decision Process (MDP). We introduce an efficient algorithm, which exploits the tree structure of the MDP, for synthesizing a control policy that maximizes the probability of satisfaction. For the proposed PCTL fragment, we define the specification update rules that guarantee the increase (or decrease) of the satisfaction probability. We introduce an incremental algorithm for synthesizing an updated MDP control policy that reuses the initial solution. The initial specification can be updated, using the rules, until the supervisor is satisfied with both the updated specification and the corresponding satisfaction probability. We propose an offline and an online application of this method.**

I. Introduction

Temporal logics, such as Linear Temporal Logic (LTL) and Computational Tree Logic (CTL), have been recently employed to express complex robot behaviors such as "go to region A and avoid region B unless regions C or D are visited" ([KGFP07], [KF08], [WTM09], [BKV10]).

In order to use existing model checking and automata game tools for motion planning (see [BK08]), many of the above-mentioned works rely on the assumption that the motion of the vehicle in the environment can be modeled as a finite system [CGP99] that is either deterministic [DLB12], nondeterministic [KB08], or probabilistic ([LAB12]). If a system is probabilistic, probabilistic temporal logics, such as Probabilistic CTL (PCTL) and Probabilistic LTL (PLTL), can be used for motion planning and control. In particular, given a robot specification expressed as a probabilistic temporal logic formula, probabilistic model checking and automata game techniques can be adapted to synthesize control policies that maximize the probability that the robot satisfies the specification ([LAB12], [CB12]).

However, in many complex tasks, it is critically important to keep humans in the loop and engaged in the overall decision-making process. Therefore, we propose a theoretical framework for a *human-supervised control synthesis method*, with an offline and online phase.

In the *offline phase* (i.e., before the deployment) the supervisor gives an initial specification and the control synthesis algorithm returns the initial satisfaction probability. If the supervisor is not satisfied with the satisfaction probability, the system generates a set of specification relaxations that guarantee an increase in the satisfaction probability. The offline phase ends when the supervisor agrees with a specification and the corresponding satisfaction probability. In the *online phase* (i.e., during the deployment), events occurring in the environment can affect the satisfaction probability. If such an event occurs, the system returns the updated control policy, and if necessary proposes an updated specification that will increase the satisfaction probability.

We focus on controlling a stochastic version of a Dubins vehicle such that the probability of satisfying a specification given as a formula in a fragment of PCTL over a set properties at the regions in the environment is maximized. We assume that the vehicle can determine its precise initial position in a known map of the environment. However, inspired by practical applications, we assume that the vehicle is equipped with noisy actuators and sensors. We extend our approach presented in [CB12] to construct a finite abstraction of the motion of the vehicle in the environment in the form of a tree-structured Markov Decision Process (MDP). For the proposed PCTL fragment, which is rich enough to express complex motion specifications, we introduce the specification update rules that guarantee the increase (or decrease) of the satisfaction probability.

We introduce two algorithms for synthesizing MDP control policies. The first provides an initial policy and the corresponding satisfaction probability and the second is used for obtaining an updated solution. In general, given an MDP and a PCTL formula, solving a synthesis problem requires solving a Linear Programing (LP) problem (see [BK08], [LAB12]). By exploiting the special tree structure of the MDP, obtained through the abstraction process, as well as the structure of the PCTL fragment, we show that our algorithms produce the optimal solution in a fast and efficient manner without solving an LP. Moreover, the second algorithm produces an updated optimal solution by reusing the initial solution. Once the MDP control policy is obtained, by establishing a mapping between the states of the MDP and the sensor measurements, the policy is mapped to a vehicle feedback control strategy.

The work presented in this paper is, to the best of our knowledge, novel. The closest related research problem is automatic formula revision for LTL specifications [Fai11], where if a specification can not be satisfied on a particular environment, the framework returns to the user an updated specification that is satisfiable. The presented work addresses

This work was partially supported by the ONR MURI under grant N00014-10-10952 and by the NSF under grant CNS-0834260.

The authors are with the Division of Systems Engineering at Boston University, Boston, MA 02215, USA. Email: {icizelj,cbelta}@bu.edu.

a different but related problem; the problem of automatic formula revision for PCTL motion planning specifications. Additionally, our framework allows for noisy actuators and sensors and for environmental changes during the deployment. Due to space limitations, preliminaries are not included in this paper. We refer readers to [BK08] for information about MDPs and to [BK08], [LAB12] for detailed description of PCTL. Furthermore, we omit all proofs of all results. An extended version of this paper can be found in [CB13].

II. Problem Formulation

Motion model: A Dubins vehicle ([Dub57]) is a unicycle with constant forward speed and bounded turning radius moving in a plane. In this paper, we consider a stochastic version of a Dubins vehicle, which captures actuator noise:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos(\theta)\\ \sin(\theta)\\ u+\varepsilon\end{bmatrix},\ u\in U, \tag{1}$$

where $(x,y)\in\mathbb{R}^2$ and $\theta\in[0,2\pi)$ are the position and orientation of the vehicle in a world frame, u is the control input (angular velocity before being corrupted by noise), U is the control constraint set, and ε is a random variable modeling the actuator noise. For simplicity, we assume that ε is uniformly distributed on the bounded interval $[-\varepsilon_{max},\varepsilon_{max}]$. However, our approach works for any continuous probability distribution supported on a bounded interval. The forward speed is normalized to 1. We denote the state of the system by $q=[x,y,\theta]^T\in SE(2)$.

Motivated by the fact that the optimal Dubins paths use only three inputs ([Dub57]), we assume $U=\{-1/\rho,0,1/\rho\}$, where ρ is the minimum turn radius. We define $W=\{u+\varepsilon|u\in U,\varepsilon\in[-\varepsilon_{max},\varepsilon_{max}]\}$ as the set of applied control inputs, i.e, the set of angular velocities that are applied to the system in the presence of noise. We assume that time is uniformly discretized (partitioned) into stages (intervals) of length Δt, where stage k is from $(k-1)\Delta t$ to $k\Delta t$. The duration of the motion is finite and it is denoted by $K\Delta t$.We denote the control input and the applied control input at stage k as $u_k\in U$ and $w_k\in W$, respectively. We assume that the noise ε is piece-wise constant, i.e, it can only change at the beginning of a stage. This implies that the applied control is also piece-wise constant, i.e., $w:[(k-1)\Delta t,k\Delta t]\to W$, $k=1,\ldots,K$, is constant over each stage.

Sensing model: We assume that the vehicle is equipped with only one sensor, which is a limited accuracy gyroscope. At stage k, the gyroscope returns the measured interval $[\underline{w}_k,\overline{w}_k]\subset[u_k-\varepsilon_{max},u_k+\varepsilon_{max}]$ containing the applied control input. Motivated by practical applications, we assume that the measurement resolution of the gyroscope, i.e., the length of $[\underline{w}_k,\overline{w}_k]$, is constant, and we denote it by $\Delta\varepsilon$. For simplicity of presentation, we also assume that $n\Delta\varepsilon=2\varepsilon_{max}$, for some $n\in\mathbb{Z}^+$. Then, $[-\varepsilon_{max},\varepsilon_{max}]$ can be partitioned[1] into n intervals: $[\underline{\varepsilon}_i,\overline{\varepsilon}_i]$, $i=1,\ldots,n$. We denote the set of all noise intervals as $\mathscr{E}=\{[\underline{\varepsilon}_1,\overline{\varepsilon}_1],\ldots,[\underline{\varepsilon}_n,\overline{\varepsilon}_n]\}$. At stage k, if the applied control input is $u_k+\varepsilon$, the gyroscope will return the measured interval $[\underline{w}_k,\overline{w}_k]=[u_k-\underline{\varepsilon},u_k+\overline{\varepsilon}]$, where $\varepsilon\in[\underline{\varepsilon},\overline{\varepsilon}]\in\mathscr{E}$. Since ε is uniformly distributed:

$$\Pr(u_k+\varepsilon\in[u_k-\underline{\varepsilon}_i,u_k+\overline{\varepsilon}_i])=\Pr(\varepsilon\in[\underline{\varepsilon}_i,\overline{\varepsilon}_i])=1/n, \tag{2}$$

$[\underline{\varepsilon}_i,\overline{\varepsilon}_i]\in\mathscr{E}$, $i=1,\ldots,n$.

Environment model: The vehicle moves in a static environment $X\subseteq\mathbb{R}^2$ in which regions of interest are present. Let Π be a finite set of propositions satisfied at the regions in the environment. Let $[\cdot]:2^{\Pi}\to 2^X$ be a map such that $[\Theta]$, $\Theta\in 2^{\Pi}$, is the set of all positions in X satisfying all and only propositions $\pi\in\Theta$. Inspired by a realistic scenario of an indoor vehicle leaving its charging station, we assume that the vehicle can precisely determine its initial state $q_{init}=[x_{init},y_{init},\theta_{init}]^T$ in a known map of the environment.

Specification: Formulas of PCTL are interpreted over states of an MDP and are constructed by connecting properties from Π using standard Boolean operators, the temporal operator $\mathscr{U}$ ("until"), and the probabilistic operator $\mathscr{P}$. In this work, we assume that the vehicle needs to carry out a motion specification expressed as a PCTL formula ϕ over Π:

$$\begin{aligned}\phi:=\mathscr{P}_{max=?}[\mathscr{P}_{\geq p_1}[\varphi_1\mathscr{U}(\psi_1\wedge\mathscr{P}_{\geq p_2}[\varphi_2\mathscr{U}(\psi_2\wedge\\ \ldots\wedge\mathscr{P}_{\geq p_f}[\varphi_f\mathscr{U}\psi_f])])]],\end{aligned} \tag{3}$$

$f\in\mathbb{Z}^+$, where $\forall j\in\{1,\ldots,f\}$, φ_j and ψ_j are PCTL formulas constructed by connecting properties from a set of propositions Π using only Boolean operators in Conjunctive Normal Form (CNF) and Disjunctive Normal Form (DNF)[2], respectively, and $p_j\in[0,1]$. We assume that ϕ is in Negation Normal Form (NNF), i.e., Boolean operator $\neg$ appears only in front of the propositions.

Example 1: Consider the environment shown in Fig. 2. Let $\Pi=\{\pi_p,\pi_{t1},\pi_{t2},\pi_{d1},\pi_{d2},\pi_u\}$, where $\pi_p,\pi_{t1},\pi_{t2},\pi_{d1},\pi_{d2},\pi_u$ label `pick-up`, `test1`, `test2`, `drop-off1`, `drop-off2` and the `unsafe` regions, respectively. Consider the following motion specification:

Specification 1: Starting form an initial state q_{init} reach a `pick-up` region, while avoiding the `test1` regions, to pick up a load. Then, reach a `test1` region or a `test2` region. Finally, reach a `drop-off1` or a `drop-off2` region to drop off the load. Always avoid the `unsafe` regions.

The specification translates to PCTL formula ϕ:

$$\begin{aligned}\phi:=\mathscr{P}_{max=?}[\mathscr{P}_{>0}[\neg\pi_u\wedge\neg\pi_{t1}\mathscr{U}(\neg\pi_u\wedge\pi_p\wedge\\ \mathscr{P}_{>0}[\neg\pi_u\mathscr{U}((\neg\pi_u\wedge\pi_{t1})\vee(\neg\pi_u\wedge\pi_{t2})\wedge\\ \mathscr{P}_{>0}[\neg\pi_u\mathscr{U}(\neg\pi_u\wedge\pi_{d1})\vee(\neg\pi_u\wedge\pi_{d2})])])]].\ \blacksquare\end{aligned} \tag{4}$$

Next, we define the satisfaction of ϕ (Eqn. 3) by a trajectory $q:[0,K\Delta t]\to SE(2)$ of the system from Eqn. (1). The formal definition is given in [CB13]. Informally, the word o produced by $q(t)$ is the sequence of sets of propositions satisfied by the position $(x(t),y(t))$ of the robot as time evolves. A trajectory $q(t)$ satisfies PCTL formula ϕ iff the corresponding sequence satisfies the formula.

[1] Throughout the paper, we relax the notion of a partition by allowing the endpoints of the intervals to overlap.

[2] A formula is CNF if it is a conjunction of clauses, where a clause is a disjunction of propositions. A formula is in DNF if it is a disjunction of clauses, where a clause is a conjunction of propositions.

As time evolves and a sequence o is generated, we can check what part of ϕ is satisfied so far. If $\mathscr{P}_{\geq p_1}[\varphi_1 \mathscr{U} (\psi_1 \wedge \ldots \wedge \mathscr{P}_{\geq p_i}[\varphi_i \mathscr{U} \psi_i]]$ part of ϕ is satisfied we say ϕ is satisfied up to i, $0 \leq i \leq f$ (for more details see Sec. IV-B). Given ϕ satisfied up to i, $0 \leq i \leq f$, the updated PCTL formula, denoted ϕ^+, is obtained from ϕ by removing the already satisfied part of ϕ, and then by 1) adding or removing conjunction clause from ψ_j, or 2) adding or removing a disjunction clause from φ_j, or 3) increasing or decreasing p_j, for any $j \in \{i, \ldots, f\}$. Formal definitions are given in Sec. IV-B. To illustrate this idea consider the following example:

Example 2: Consider Specification 1 and assume that at $k\Delta t$ the vehicle enters a `pick-up` region, while avoiding the `test1` and the `unsafe` regions, and additionally, that the `drop-off2` regions become unavailable for the drop off, i.e., the vehicle is allowed to drop off the load only at the `drop-off1` regions. Then, the updated formula is:

$$\phi^+ := \mathscr{P}_{max=?}[\mathscr{P}_{>0}[\neg\pi_u \mathscr{U} ((\neg\pi_u \wedge \pi_{t1}) \vee (\neg\pi_u \wedge \pi_{t2}) \wedge \mathscr{P}_{>0}[\neg\pi_u \mathscr{U} (\neg\pi_u \wedge \pi_{d1})])]],$$

where ϕ^+ is obtained from ϕ by removing the already satisfied part of ϕ, $\mathscr{P}_{>0}[\neg\pi_u \wedge \neg\pi_{t1} \mathscr{U} \neg\pi_u \wedge \pi_p]$, and by removing the conjunction clause, $(\neg\pi_u \wedge \pi_{d2})$, from ψ_3. ■

While the vehicle moves, gyroscope measurements $[\underline{w}_k, \overline{w}_k]$ are available at each stage k. We define a *vehicle control strategy* as a map that takes as input a sequence of measured intervals $[\underline{w}_1, \overline{w}_1][\underline{w}_2, \overline{w}_2] \ldots [\underline{w}_{k-1}, \overline{w}_{k-1}]$ and returns the control input $u_k \in U$ at stage k. We are ready to formulate the main problem that we consider in this paper:

Problem 1: Given a set of regions of interest in environment $X \subseteq \mathbb{R}^2$ satisfying propositions from set Π, a vehicle model described by Eqn. (1) with initial state q_{init}, an initial and updated motion specifications, expressed as PCTL formulas ϕ and ϕ^+, respectively, over Π (Eqn. (3)), find a vehicle control strategy that maximizes the probability of satisfying ϕ and then ϕ^+.

III. Construction of an MDP Model

In order to use the abstraction process from [CB12], we need to transform the input formula ϕ into a formula where the propositions are classified into two nonintersecting sets according to whether they represent regions that must be reached or avoided. Details are given in [CB13]. In short, we introduce an extended set of propositions $\Xi_\Pi = \Xi_\Pi^+ \cup \Xi_\Pi^-$ and a new map $[\cdot]^{\Xi_\Pi} : \Xi_\Pi \to 2^X$ for the interpretation of the propositions. We translate ϕ into a formula $\boldsymbol{\phi}$ where the occurrences of terms π and $\neg\pi$ have been replaced by the members $\xi_\pi \in \Xi_\Pi^+$ and $\xi_{\neg\pi} \in \Xi_\Pi^-$, respectively. This allows us to distinguish the regions that must be avoided (Ξ_Π^-) and the regions that must be reached (Ξ_Π^+). We show that ϕ is equivalent to $\boldsymbol{\phi}$ under the maps $[\cdot] : \Pi \to 2^X$ and $[\cdot]^{\Xi_\Pi} : \Xi_\Pi \to 2^X$. Thus, next results are given with respect to a formula $\boldsymbol{\phi}$ and a map $[\cdot]^{\Xi_\Pi} : \Xi_\Pi \to 2^X$ (see Fig. 1 for more details).

A. Approximation

We use $q_k(t)$ and w_k, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$ to denote the state trajectory and the constant applied control at stage k, respectively. With a slight abuse of notation, we use q_k to denote the end of state trajectory $q_k(t)$, i.e., $q_k = q_k(k\Delta t)$. Given a state q_{k-1}, the state trajectory $q_k(t)$ can be derived by integrating the system given by Eqn. (1) from the initial state q_{k-1}, and taking into account that the applied control is constant and equal to w_k. Throughout the paper, we will also denote this trajectory by $q_k(q_{k-1}, w_k, t)$, when we want to explicitly capture the initial state q_{k-1} and the constant applied control w_k.

For each interval in $\mathscr{E}$ we define a representative value $\varepsilon_i = \frac{\underline{\varepsilon}_i + \overline{\varepsilon}_i}{2}$, $i = 1, \ldots, n$. i.e., ε_i is the midpoint of interval $[\underline{\varepsilon}_i, \overline{\varepsilon}_i]$. We denote the set of all representative values as $E = \{\varepsilon_1, \ldots, \varepsilon_n\}$. We define $W_d = \{u + \varepsilon \mid u \in U, \varepsilon \in E\} \subset W$ as a finite set of applied control inputs. Also, let $\omega : U \to W_d$ be a random variable, where $\omega(u) = u + \varepsilon$ with the probability mass function $p_\omega(\omega(u) = u + \varepsilon) = \frac{1}{n}$ (follows from Eqn. (2)).

Finally, we define a Quantized System (QS) that approximates the original system as follows: The set of applied control inputs in QS is W_d; for a state q_{k-1} and a control input $u_k \in U$, QS returns $q_k(q_{k-1}, \omega(u_k), t) = q_k(q_{k-1}, u_k + \varepsilon, t)$ with probability $\frac{1}{n}$, where $\varepsilon \in E$.

Next, we denote $u_1 u_2 \ldots u_K$, in which $u_k \in U$ gives a control input at stage k, as a finite sequence of control inputs of length K. Let Σ_K denote the set of all such sequences. For the initial state q_{init} and Σ_K, we define the reachability graph $G_K(q_{init})$ (see [LaV06] for a related definition), which encodes the set of all state trajectories originating from q_{init} that can be obtained, with a positive probability, by applying sequences of control inputs from Σ_K according to QS.

B. Position uncertainty and MDP construction

The position uncertainty of the vehicle when its nominal position is $(x, y) \in \mathbb{R}^2$ is modeled as a disc centered at (x, y) with radius $r \in \mathbb{R}$, where r denotes the uncertainty: $D((x, y), r) = \{(x', y') \in \mathbb{R}^2 \mid\mid (x, y), (x', y')\mid\mid \leq r\}$. The way we model the uncertainty along $q(t) \in G_K(q_{init})$ is given in [CB12]. Briefly, first, we obtain uncertainty at state q_k, denoted r_k, by using a worst case scenario assumption: if $u_k + \varepsilon_k \in W_d$ is the applied control input for QS, the corresponding applied control input at stage k for the original system was $u_k - \underline{\varepsilon}_k$ or $u_k + \overline{\varepsilon}_k$, where $\varepsilon_k \in [\underline{\varepsilon}_k, \overline{\varepsilon}_k]$. Then, we define $r : [0, K\Delta t] \to \mathbb{R}$ as an approximated uncertainty trajectory and we set $r(t) = r_k$, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$, i.e., we set the uncertainty along the state trajectory $q_k(t)$ equal to the maximum value of the uncertainty along $q_k(t)$, which is at state q_k (for more details see Fig. 1).

A tree-structured MDP M that models the motion of the vehicle in the environment and the evolution of the position uncertainty is defined as a tuple $(S, s_0, Act, A, P, \Xi_\Pi, h)$ where:

- S is the finite set of states. The meaning of the state is as follows: $(q(t), r(t), \underline{\varepsilon}, \overline{\varepsilon}, \Theta) \in S$ means that along the state trajectory $q(t)$, the uncertainty trajectory is $r(t)$; the noise interval is $[\underline{\varepsilon}, \overline{\varepsilon}] \in \mathscr{E}$; and $\Theta \in 2^{\Xi_\Pi}$ is the set of satisfied propositions along the state trajectory $q(t)$ when $r(t)$ is the uncertainty trajectory (see Fig. 1 for an example).
- $s_0 = (q_{init}, 0, 0, 0, \Theta_{init}) \in S$ is the initial state;
- $Act = U \cup \nu$ is the set of actions (ν is a dummy action);
- $A : S \to 2^{Act}$ gives the enabled actions at each state;

• $P : S \times Act \times S \to [0,1]$ is a transition probability function such that a) for all states $s \in S$ and actions $a \in A(s)$: $\sum_{s' \in S} P(s,a,s') = 1$, b) for all actions $a \notin A(s)$ and $s' \in S$, $P(s,a,s') = 0$, and c) for all states $s \in S \setminus s_0$ there exists exactly one state−action pair $(s',a) \in S \times A(s')$, s.t. $P(s',a,s) > 0$. In other words in a tree-structured MDP, each state has only one incoming transition, i.e., there are no cycles.

• Ξ_π is the set of propositions;

• $h : S \to 2^{\Xi_\pi}$ assigns proposition from Ξ_Π to states $s \in S$ according to the following rule: given $s = (q(t), r(t), \underline{\varepsilon}, \overline{\varepsilon}, \Theta) \in S$, $\forall \xi \in \Xi_\Pi$, $\xi \in h(s)$ iff $\xi \in \Theta$.

We generate S and P while building $G_K(q_{init})$ starting from q_{init}. Given $q_k(t) = q_k(q_{k-1}, u_k + \varepsilon, t) \in G_K(q_{init})$, and the corresponding $r_k(t)$, $t \in [(k-1)\Delta t, k\Delta t]$, $k = 1, \ldots, K$, first, we generate a sequence $(\Theta_k^1, [\underline{t}_k^1, \overline{t}_k^1]), \ldots, (\Theta_k^l, [\underline{t}_k^l, \overline{t}_k^l])$, $l \geq 1$, where $\Theta_k^i \in 2^{\Xi_\Pi}$ is the set of satisfied propositions along the state trajectory $q_k^i(t) = q_k(t')$, when the corresponding uncertainty trajectory is $r_k^i(t) = r_k(t')$, for $t' \in [\underline{t}_k^i, \overline{t}_k^i] \subseteq [(k-1)\Delta t, k\Delta t]$, $i = 1, \ldots, l$, according to the following rules:

• Let $\underline{t}_k^1 = (k-1)\Delta t$. Then, $D((x_k(\underline{t}_k^1), y_k(\underline{t}_k^1)), r_k(\underline{t}_k^1)) \subseteq [\Theta_k^1]$ and $\overline{t}_k^1 = \max_{[\underline{t}_k^1, k\Delta t]} \{t | D((x_k(t), y_k(t), r_k(t) \subseteq [\Theta_k^1]\}$.

• If $D((x_k(t), y_k(t), r_k(t)) \subseteq [\Theta_k^i]$, $t \in [\underline{t}_k^i, \overline{t}_k^i]$ and $\Theta_k^{i+1} \neq \Theta_k^i$, then:

1) $\exists t \geq \overline{t}_k^i$ s.t. $D((x_k(t), y_k(t), r_k(t)) \subseteq [\Theta_k^{i+1}]$,

2) $D((x_k(\tau), y_k(\tau), r_k(\tau)) \not\subseteq [\xi]$, $\forall \tau \in [\overline{t}_k^i, t]$, $\forall \xi \in \Xi_\Pi \setminus (\Theta_k^i \cup \Theta_k^{i+1})$ and

3) $\underline{t}_k^{i+1} = \overline{t}_k^i$ and $\overline{t}_k^{i+1} = \max_{[\underline{t}_k^{i+1}, k\Delta t]} \{t | D((x_k(t), y_k(t), r_k(t) \subseteq [\Theta_k^{i+1}]\}$.

Next, for each $(\Theta_k^i, [\underline{t}_k^i, \overline{t}_k^i])$, $i = 1, \ldots, l$, we generate a state of the MDP $s_k^i = (q_k^i(t), r_k^i(t), \underline{\varepsilon}, \overline{\varepsilon}, \Theta_i^k)$ such that $q_k^i(t) = q_k(t')$ and $r_k^i(t) = r_k(t')$, $t' \in [\underline{t}_k^i, \overline{t}_k^i]$ and $\underline{\varepsilon}$ and $\overline{\varepsilon}$ are such that $\varepsilon \in [\underline{\varepsilon}, \overline{\varepsilon}] \in \mathscr{E}$. Finally, the newly generated state s_k^i, $i = 1, \ldots l$, $l \geq 1$, is added to S and the transition probability function is updated, as follows:

• If $i < l$, $A(s_k^i) = \nu$ and $P(s_k^i, \nu, s_k^{i+1}) = 1$, and otherwise, i.e., if $i = l$, $A(s_k^l) = U$ and $\forall u_{k+1} \in U$, $P(s_k^l, u_{k+1}, s_{k+1}^1) = \frac{1}{n}$.

The former follows from the fact that $k\Delta$ is not reached and control input for the next stage needs not to be chosen. Under dummy action ν, with probability 1, the system makes a transition to the next state in the sequence satisfying a different set of propositions. The latter follows from the fact that $k\Delta t$ is reached and the control input for the next stage needs to be chosen.

• If the termination time is reached, we set $A(s_k^i) = \nu$ and $P(s_k^i, \nu, s_k^i) = 1$. Such state is called a *leaf* state. In [CB13] we show that M is a valid tree-structured MDP.

IV. PCTL Control Policy Generation

A. Control policy for the initial PCTL formula

The proposed PCTL control synthesis is an adaptation of the approach from [LAB12]. Given a tree-structured MDP $M = (S, s_0, A, Act, P, \Xi_\Pi, h)$ and a PCTL formula $\phi := \mathscr{P}_{max=?}[\mathscr{P}_{\geq p_1}[\varphi_1 \mathscr{U} (\psi_1 \wedge \mathscr{P}_{\geq p_2}[\varphi_2 \mathscr{U} (\psi_2 \wedge \ldots \wedge \mathscr{P}_{\geq p_f}[\varphi_f \mathscr{U} \psi_f])])]]$, we are interested in obtaining the control policy μ_ϕ that maximizes the probability of satisfying ϕ, as well as the corresponding probability value, denoted

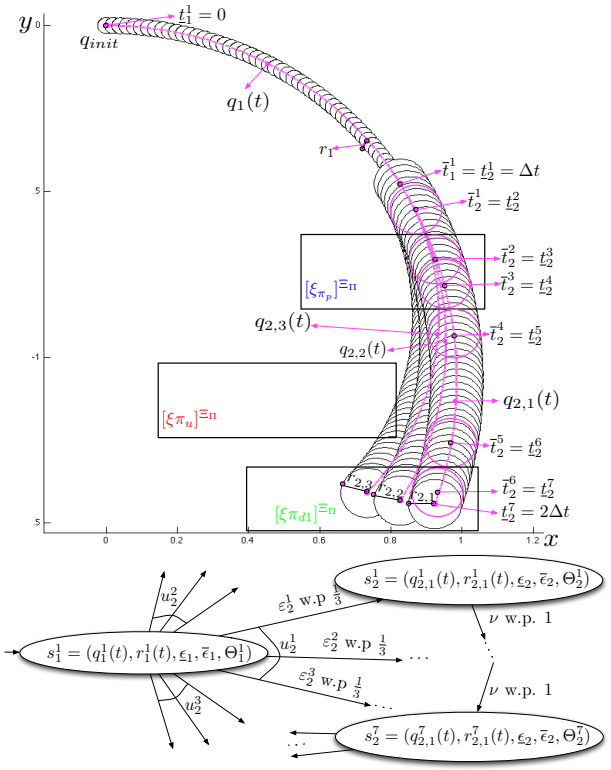

Fig. 1. Above: An example scenario corresponding to the MDP fragment shown below. $[\xi_{\pi_u}]^{\Xi_\Pi}$, $[\xi_{\pi_p}]^{\Xi_\Pi}$ and $[\xi_{\pi_{d1}}]^{\Xi_\Pi}$ are shown in the figure. Then $[\xi_{\neg\pi_u}]^{\Xi_\Pi} = X \setminus [\xi_{\pi_u}]$, and similarly for $[\xi_{\neg\pi_p}]^{\Xi_\Pi}$ and $[\xi_{\neg\pi_{d1}}]^{\Xi_\Pi}$ holds. Since along the state trajectory $q_1(t)$ when the uncertainty trajectory is $r_1(t) = r_1$, $t \in [0, \Delta t]$ the set of satisfying propositions does not change, only one state, denoted s_1^1, is generated, where $\theta_1 = \{\xi_{\neg\pi_u}, \xi_{\neg\pi_p}, \xi_{\neg\pi_{d1}}\}$. For the state trajectory $q_{2,1}(t)$, when the uncertainty trajectory is $r_{2,1}(t) = r_{2,1}$, $t \in [\Delta t, 2\Delta t]$, the following sequence is generated: $(\Theta_2^1, [\underline{t}_2^1, \overline{t}_2^1]), \ldots, (\Theta_2^7, [\underline{t}_2^7, \overline{t}_2^7])$, where the time interval bounds are shown on the figure and $\Theta_2^1 = \{\xi_{\neg\pi_u}, \xi_{\neg\pi_p}, \xi_{\neg\pi_{d1}}\}$, $\Theta_2^2 = \{\xi_{\neg\pi_u}, \xi_{\neg\pi_{d1}}\}$, $\Theta_2^3 = \{\xi_{\neg\pi_u}, \xi_{\pi_p}, \xi_{\neg\pi_{d1}}\}$, …, $\Theta_2^6 = \{\xi_{\neg\pi_u}, \xi_{\neg\pi_p}\}$ and $\Theta_2^7 = \{\xi_{\neg\pi_u}, \xi_{\neg\pi_p}, \xi_{\pi_{d1}}\}$. Below: A fragment of the MDP corresponding to the scenario shown above, where $[-\varepsilon_{max}, \varepsilon_{max}]$ is partitioned into $n = 3$ intervals. Action $u_2^1 \in A(s_1^1)$ enables three transitions, each w.p. $\frac{1}{3}$. This corresponds to applied control input being equal to $u_2^1 + \varepsilon_2^i$ w.p. $\frac{1}{3}$, $\varepsilon_2^i \in E$. The elements of s_2^i are: $q_2^i(t) = q_{2,1}(t')$ and $r_2^i(t) = r_{2,1}(t')$, $t' \in [\underline{t}_2^i, \overline{t}_2^i]$, $[\underline{\varepsilon}_2, \overline{\varepsilon}_2]$ is such that $\varepsilon_2^1 \in [\underline{\varepsilon}_2, \overline{\varepsilon}_2] \in \mathscr{E}$ and Θ_2^i, $i = 1, \ldots, 7$.

V_ϕ, where $V_\phi : S \to [0,1]$. Specifically, for $s \in S$, $\mu_\phi(s) \in A(s)$ is the action to be applied at s and $V_\phi(s)$ is the probability of satisfying ϕ at s under control policy μ_ϕ. To solve this problem we propose the following approach:

Step 1: Solve $\phi_f := \mathscr{P}_{\geq p_f}[\varphi_f \mathscr{U} \psi_f]$, i.e., find the set of initial states S_{ϕ_f} from which ϕ_f is satisfied with probably greater than or equal to p_f and determine the corresponding control policy μ_{ϕ_f}. To solve this problem, first, let $\phi_f' := \mathscr{P}_{\max=?}[\varphi_f \mathscr{U} \psi_f]$, and compute the maximizing probabilities $V_{\phi_f'}$. This can be done by dividing S into three subsets $S_{\phi_f'}^{yes}$ (states satisfying ϕ_f' with probability 1), $S_{\phi_f'}^{no}$ (states satisfying ϕ_f' with probability 0), and $S_{\phi_f'}^{?}$ (the remaining states): $S_{\phi_f'}^{yes} = \text{Sat}(\psi_f)$, $S_{\phi_f'}^{no} = S \setminus (\text{Sat}(\varphi_f) \cup \text{Sat}(\psi_f))$ and $S_{\phi_f'}^{?} = S \setminus (S_{\phi_f'}^{yes} \cup S_{\phi_f'}^{no})$, where $\text{Sat}(\psi_f)$ and $\text{Sat}(\varphi_f)$ are the set of states satisfying ψ_f and φ_f, respectively. The computation

of maximizing probabilities for the states in S can be obtained as a unique solution of the following system:

$$V_{\boldsymbol{\phi}'_f}(s) = \begin{cases} 1 & \text{if} \quad s \in S^{yes}_{\boldsymbol{\phi}'_f} \\ 0 & \text{if} \quad s \in S^{no}_{\boldsymbol{\phi}'_f} \\ \max_{a \in A(s)}\{\sum_{s' \in S} P(s,a,s') V_{\boldsymbol{\phi}'_f}(s')\} & \text{if} \quad s \in S^{?}_{\boldsymbol{\phi}'_f} \end{cases} \tag{5}$$

and the control policy at each state is equal to the action that gives rise to this optimal solution, i.e., $\forall s \in S$, $\mu_{\boldsymbol{\phi}'_f}(s) = \text{argmax}_{a \in A(s)}\{\sum_{s' \in S} P(s,a,s') V_{\boldsymbol{\phi}'_f}(s')\}$.

In general (i.e., for a non tree-structured MDPs containing cycles), solving Eqn. (5) requires solving a linear programming problem ([BK08], [LAB12]). For a tree-structured MDPs the solution can be obtained in a simple fashion: from each leaf state of the MDP, move backwards, by visiting parent states until s_0 is reached; at each state in $S^{?}_{\boldsymbol{\phi}'_f}$ perform maximization from Eqn. (5). The fact that M contains no cycles is sufficient to see that the procedure stated above will result in maximizing probabilities.

The state formula $\boldsymbol{\phi}_f$ requires to reach a state in $\text{Sat}(\psi_f)$ by going through states in $\text{Sat}(\varphi_f)$ with probability greater than or equal to p_f. Thus, $\forall s \in S$ s.t. $V_{\boldsymbol{\phi}'_f}(s) < p_f$ we set $V_{\boldsymbol{\phi}_f}(s) = 0$, and otherwise, i.e., $\forall s \in S$ s.t. $V_{\boldsymbol{\phi}'_f}(s) \geq p_f$ we set $V_{\boldsymbol{\phi}_f}(s) = V_{\boldsymbol{\phi}'_f}(s)$. Finally, $\forall s \in S$, $\mu_{\boldsymbol{\phi}_f}(s) = \mu_{\boldsymbol{\phi}'_f}(s)$ and the set of initial states is $S_{\boldsymbol{\phi}_f} = \{s \in S | V_{\boldsymbol{\phi}_f}(s) > 0\}$.

Step 2: Solve $\boldsymbol{\phi}_{f-1} := \mathscr{P}_{\geq p_{f-1}}[\varphi_f \mathscr{U} (\psi_{f-1} \wedge \boldsymbol{\phi}_f)]$, i.e., find the set of initial states $S_{\boldsymbol{\phi}_{f-1}}$ from which $\boldsymbol{\phi}_{f-1}$ is satisfied with probability greater than or equal to p_{f-1}. To solve this problem, again, begin by solving $\boldsymbol{\phi}'_{f-1} := \mathscr{P}_{\max=?}[\varphi_{f-1} \mathscr{U} (\psi_{f-1} \wedge \boldsymbol{\phi}_f)]$. Start by dividing S into three subsets: $S^{yes}_{\boldsymbol{\phi}'_{f-1}} = \text{Sat}(\psi_{f-1}) \cap S_{\boldsymbol{\phi}_f}$, $S^{no}_{\boldsymbol{\phi}'_{f-1}} = S \setminus (\text{Sat}(\varphi_{f-1}) \cup S^{yes}_{\boldsymbol{\phi}'_{f-1}}))$ and $S^{?}_{\boldsymbol{\phi}'_{f-1}} = S \setminus (S^{yes}_{\boldsymbol{\phi}'_{f-1}} \cup S^{no}_{\boldsymbol{\phi}'_{f-1}})$. Note that, $S^{yes}_{\boldsymbol{\phi}'_{f-1}}$ is the set of states satisfying ψ_{f-1} intersected with $S_{\boldsymbol{\phi}_f}$. Next, perform the same procedure as in *Step 1* for obtaining $V_{\boldsymbol{\phi}_{f-1}}$, $\mu_{\boldsymbol{\phi}_{f-1}}$ and $S_{\boldsymbol{\phi}_{f-1}}$.

Step 3: Repeat *Step 2* for $\boldsymbol{\phi}_{f-2}, \boldsymbol{\phi}_{f-3}, \ldots, \boldsymbol{\phi}_1$.

By the nature of the PCTL formulas, to ensure the execution of all specified tasks in $\boldsymbol{\phi}$, we construct a history dependent control policy $\mu_{\boldsymbol{\phi}}$: *Apply policy* $\mu_{\boldsymbol{\phi}_1}$ *until a state in* $S^{yes}_{\boldsymbol{\phi}'_1}$ *is reached. Then, apply policy* $\mu_{\boldsymbol{\phi}_2}$ *until a state in* $S^{yes}_{\boldsymbol{\phi}'_2}$ *is reached.* $\cdots$ *Finally, apply* $\mu_{\boldsymbol{\phi}_f}$ *until a state in* $S^{yes}_{\boldsymbol{\phi}'_f}$ *is reached.* For the same reason we can only find the lower and upper bounds of $V_{\boldsymbol{\phi}}(s_0)$. The lower and upper bounds are $V_{\boldsymbol{\phi}_1}(s_0) \cdot V^{min}_{\boldsymbol{\phi}_2} \cdot \ldots \cdot V^{min}_{\boldsymbol{\phi}_f}$ and $V_{\boldsymbol{\phi}_1}(s_0) \cdot V^{max}_{\boldsymbol{\phi}_2} \cdot \ldots \cdot V^{max}_{\boldsymbol{\phi}_f}$, where $V^{min}_{\boldsymbol{\phi}_i}$ and $V^{max}_{\boldsymbol{\phi}_i}$ are the minimum and maximum probability of satisfying $\boldsymbol{\phi}_i$ from $S^{yes}_{\boldsymbol{\phi}'_{i-1}}$.

In [CB12] we show that a sequence of measured intervals corresponds to a unique state of the MDP. The vehicle control strategy maps the sequence to the state of the MDP and returns the control input for the next stage corresponding to the to the optimal action, under $\mu_{\boldsymbol{\phi}}$, at that state.

B. Control policy for the updated PCTL formula

Next, assume that at the end of stage k, for some $k = 0, \ldots, K-1$, $\boldsymbol{\phi}$ is updated into $\boldsymbol{\phi}^+$. As noted in the previous subsection, given a sequence of measured intervals, we can follow vehicle's progress on M. We denote the current state as $s_C \in S$ (if it is at the initial state, then $s_C = s_0$). We develop an efficient algorithm for obtaining $\mu_{\boldsymbol{\phi}^+}$, and $V_{\boldsymbol{\phi}^+}$, that reuses $\mu_{\boldsymbol{\phi}}$ and $V_{\boldsymbol{\phi}}$, and exploits the structure of formulas given by Eqn. (3) and the fact that M is a tree-structured MDP.

First, we formally define what it means for $\boldsymbol{\phi}$ to be satisfied up to i, $0 \leq i \leq f$. Note that, if under the execution of $\mu_{\boldsymbol{\phi}}$, $S^{yes}_{\boldsymbol{\phi}'_i}$ is reached, it is guaranteed that $\mathscr{P}_{\geq p_1}[\varphi_1 \mathscr{U} (\psi_1 \wedge \ldots \wedge \mathscr{P}_{\geq p_i}[\varphi_i \mathscr{U} \psi_i])]$ part of $\boldsymbol{\phi}$ is satisfied. Thus, $\boldsymbol{\phi}$ is satisfied up to i, where $i = \max_{j \in \{0,\ldots,f\}}\{j | S^{yes}_{\boldsymbol{\phi}'_j}$ is reached$\}$, $S^{yes}_{\boldsymbol{\phi}'_0} = s_0$. Next, since $\forall j \in \{1, \ldots, f\}$, φ_j and ψ are in CNF and DNF, respectively, they can be expressed as $\varphi_j = \varphi^1_j \wedge \ldots \wedge \varphi^{m_j}_j$ and $\psi_j = \psi^1_j \vee \ldots \vee \psi^{n_j}_j$ where $m_j, n_j \in \mathbb{Z}^+$ and $\forall_{m=1,\ldots,m_j} \varphi^m_j$ is a disjunction clause and $\forall_{n=1,\ldots,n_j} \psi^n_j$ is a conjunction clause.

Specification update rules: Given $\boldsymbol{\phi}$ satisfied up to i, $0 \leq i \leq f$, the updated formula $\boldsymbol{\phi}^+$ is obtained from $\boldsymbol{\phi}$ by removing $\mathscr{P}_{\geq p_1}[\varphi_1 \mathscr{U} (\psi_1 \wedge \ldots \wedge \mathscr{P}_{\geq p_i}[\varphi_i \mathscr{U} \psi_i])]$ from $\boldsymbol{\phi}$, and then by updating ψ_j, φ_j, or p_j for $j \in \{i, \ldots, f\}$:

1) $\psi^+_j = \psi^1_j \vee \ldots \vee \psi^{n_j+1}_j$; or
2) $\psi^+_j = \psi^1_j \vee \ldots \vee \psi^{n_j-1}_j$, if $n_j \geq 1$; or
3) $\varphi^+_j = \varphi^1_j \wedge \ldots \wedge \varphi^{m_j-1}_j$, if $m_j \geq 1$; or
4) $\varphi^+_j = \varphi^1_j \wedge \ldots \wedge \varphi^{m_j+1}_j$; or
5) $p^+_j \in [0,1]$ s.t. $p^+_j < p_j$; or
6) $p^+_j \in [0,1]$ s.t. $p^+_j > p_j$; where $\psi^{n_j+1}_j$ and $\varphi^{m_j-1}_j$ are conjunction and disjunction clauses from Ξ_Π, respectively.

First, note that since M is a tree-structured MDP, $\mu_{\boldsymbol{\phi}^+}$ needs to be defined only for the states reachable from current state $s_C \in S$. Thus, we construct a new tree-structured MDP $M^+ \subseteq M$, for which s_C is the initial state, by eliminating the states that are not reachable form s_C. For a tree-structured MDP this is a straightforward process. By using the approach presented in Sec. IV-A we can partially reuse $\mu_{\boldsymbol{\phi}}$ and $V_{\boldsymbol{\phi}}$ when solving the problem. Additionally, the following holds: for updates 1, 3, and 5, $\forall s \in S^+$, $V_{\boldsymbol{\phi}^+}(s) \geq V_{\boldsymbol{\phi}}(s)$, and for updates 2, 4, and 6, $\forall s \in S^+$, $V_{\boldsymbol{\phi}^+}(s) \leq V_{\boldsymbol{\phi}}(s)$, where S^+ is the set of states of M^+. In this paper, due to space limitations, we prove the statements above only for *Update 1*. For the rest of the updates see the technical report [CB13].

Update 1: Since for $k \in \{j+1, \ldots, f\}$, $\boldsymbol{\phi}^+_k = \boldsymbol{\phi}_k$, it follows that $\mu_{\boldsymbol{\phi}^+_k} = \mu_{\boldsymbol{\phi}_k}$, $V_{\boldsymbol{\phi}^+_k} = V_{\boldsymbol{\phi}_k}$, and $S_{\boldsymbol{\phi}^+_k} = S_{\boldsymbol{\phi}_k}$ (this holds for all other updates as well). When solving $\boldsymbol{\phi}^+_j := \mathscr{P}_{\geq p_j}[\varphi_j \mathscr{U} ((\psi_j \vee \psi^{n_j+1}_j) \wedge \boldsymbol{\phi}_{j+1})]$, i.e., in particular $\boldsymbol{\phi}^{+\prime}_j := \mathscr{P}_{\max=?}[\varphi_j \mathscr{U} ((\psi_j \vee \psi^{n_j+1}_j) \wedge \boldsymbol{\phi}_{j+1})]$ note that: $S^{yes}_{\boldsymbol{\phi}^{+\prime}_j} = (\text{Sat}(\psi_j) \cap S_{\boldsymbol{\phi}_{j+1}}) \cup (\text{Sat}(\psi^{n_j+1}_j) \cap S_{\boldsymbol{\phi}_{j+1}})$, $S^{no}_{\boldsymbol{\phi}^{+\prime}_j} = S \setminus (\text{Sat}(\varphi_j) \cup S^{yes}_{\boldsymbol{\phi}^{+\prime}_j}))$ and $S^{?}_{\boldsymbol{\phi}^{+\prime}_j} = S \setminus (S^{yes}_{\boldsymbol{\phi}'_j} \cup S^{no}_{\boldsymbol{\phi}'_j})$. By using Eqn. (5) we obtain $\mu_{\boldsymbol{\phi}^{+\prime}_j}$ and $V_{\boldsymbol{\phi}^{+\prime}_j}$, and then $\mu_{\boldsymbol{\phi}^+_j}$, $V_{\boldsymbol{\phi}^+_j}$ and $S_{\boldsymbol{\phi}^+_j}$ as described in Sec. IV-A. From the fact that $S^{yes}_{\boldsymbol{\phi}^{+\prime}_j} \supseteq S^{yes}_{\boldsymbol{\phi}'_j}$ it follows that $\forall s \in S^+$, $V_{\boldsymbol{\phi}^{+\prime}_j}(s) \geq V_{\boldsymbol{\phi}'_j}(s)$, and thus $S_{\boldsymbol{\phi}^+_j} \supseteq S_{\boldsymbol{\phi}_j}$ and $V_{\boldsymbol{\phi}^+_j}(s) \geq V_{\boldsymbol{\phi}_j}(s)$. This property holds all the way down until $\mu_{\boldsymbol{\phi}^+_{i+1}}$ and $V_{\boldsymbol{\phi}^+_{i+1}}$ are obtained. Therefore, $\forall s \in S^+$, $V_{\boldsymbol{\phi}^+}(s) \geq V_{\boldsymbol{\phi}}(s)$. For the reasons stated in the

previous subsection μ_{ϕ^+} has also a history dependent form and we can find the lower and upper bounds of $V_{\phi^+}(s_C)$.

V. Case Study

We considered the system given by Eqn. (1) and we used the following numerical values: $1/\rho = \pi/3$, $\Delta t = 1.2$, $K = 9$, and $\varepsilon_{max} = 0.06$ with $n = 3$, i.e., $\Delta\varepsilon = 0.04$. Thus, the maximum actuator noise was approximately 6% of the maximum control input. Three cases are shown in Fig. 2.

Offline phase: Cases *A* and *B* correspond to the offline phase. Initially, the motion specification was as given in Example 1, and the corresponding PCTL formula was ϕ (Eqn. 4). The lower bound on the probability of satisfying ϕ on the corresponding MDP was 0.68. For case *A* we assumed that the supervisor was satisfied with the satisfaction probability and the vehicle was deployed under the obtained vehicle control strategy. Case *B* corresponds to the case when the user is not satisfied with a satisfaction probability of 0.68. Then, the system generated a set of specification relaxations, based on the specification update rules from Sec. IV-B, that guaranteed an increase in the satisfaction probability. We assumed that the supervisor agreed with the specification which "allowed the vehicle to go through a `test1` region before entering a `pick-up` region" (corresponds to *Update 3*), with the corresponding satisfaction probability being 0.85 (*Update 3* increases the satisfaction probability).

Online phase: Case *C* corresponds to the online phase. The vehicle was deployed under the initial vehicle control strategy from case *A* and at $5\Delta t$ the `drop-off2` regions became unavailable for the drop off, and thus the updated specification "allowed the vehicle to drop off the load only in the `drop-off1` regions" (corresponds to *Update 2*). The updated satisfaction probability, returned by the the control synthesis part, was 0.63 (*Update 2* reduces the satisfaction probability). Assuming that the supervisor was satisfied with the updated satisfaction probability the vehicle continued the deployment, now under the updated vehicle control strategy.

The constructed MDP had approximately 45000 states. The Matlab code used to construct the MDP ran for 8 min and 52 sec. The control synthesis algorithm for case *A* (initial PCTL control policy generation) ran for 23 sec. For cases *B* and *C* (updated PCTL control policy generation) the control synthesis algorithm ran for 11 and 6 seconds, respectively. In cases *B* and *C* the running time improved by reusing the initial solution from case *A*. There was an additional improvement in case *C* since the vehicle was moving prior to the update and the updated solution was obtained on the reduced MDP.

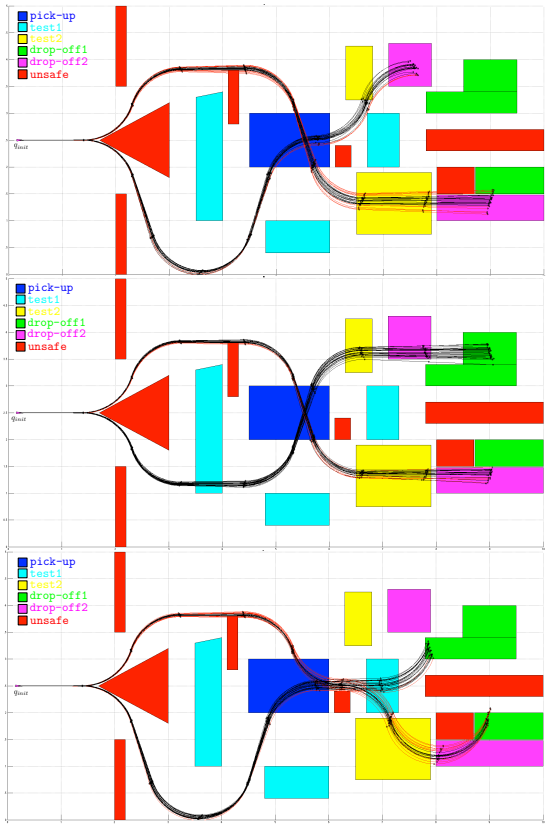

Fig. 2. 50 sample state (position) trajectories for cases *A*, *B* and *C* (to be read top to bottom) obtained by simulating the original system under the corresponding vehicle control strategies. Satisfying and violating trajectories are shown in black and red, respectively.

References

[BK08] C. Baier and J. P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BKV10] A. Bhatia, L.E. Kavraki, and M.Y. Vardi. Sampling-based Motion Planning with Temporal Gaoals. In *International Conference on Robotics and Automation (ICRA) 2010*, 2010.

[CB12] I. Cizelj and C. Belta. Probabilistically Safe Control of Noisy Dubins Vehicles. In *IEEE Intelligent Robots and Systems (IROS) Conference, 2012*, pages 2857 –2862, October 2012.

[CB13] I. Cizelj and C. Belta. Negotiating the Probabilistic Satisfaction of Temporal Logic Motion Specifications. Technical report, 2013. (available online at http://arxiv.org/abs/1307.3224).

[CGP99] E. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[DLB12] X. C. Ding, M. Lazar, and C. Belta. Receding Horizon Temporal Logic Control for Finite Deterministic Systems. In *American Control Conference (ACC) 2012*, June 2012.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[Fai11] G. Fainekos. Revising Temporal Logic Specifications for Motion Planning. In *nternational Conference on Robotics and Automation (ICRA) 2011*, pages 40–45, 2011.

[KB08] M. Kloetzer and C. Belta. Dealing with Non-Determinism in Symbolic Control. In *Hybrid Systems: Computation and Control (HSCC) 2008*, pages 287–300, 2008.

[KF08] S. Karaman and E. Frazzoli. Vehicle Routing Problem with Metric Temporal Logic Specifications. In *Conference on Decision and Control (CDC) 2008*, pages 3953 –3958, 2008.

[KGFP07] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Where's Waldo? Sensor-Based Temporal Logic Motion Planning. In *International Conference on Robotics and Automation (ICRA) 2007*, pages 3116–3121, 2007.

[LAB12] M. Lahijanian, S. B. Andersson, and C. Belta. Temporal Logic Motion Planning and Control With Probabilistic Satisfaction Guarantees. *IEEE Transactions on Robotics*, 28(2), 2012.

[LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[WTM09] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding Horizon Temporal Logic Planning for Dynamical Systems. In *Conference on Decision and Control (CDC) 2009*, 2009.