

# Specification Revision for Markov Decision Processes with Optimal Trade-off

M. Lahijanian and M. Kwiatkowska

**Abstract**—Optimal control policy synthesis for probabilistic systems from high-level specifications is increasingly often studied. One major question that is commonly faced, however, is what to do when the optimal probability of achieving the specification is not satisfactory? We address this question by viewing the specification as a soft constraint and present a synthesis framework for MDPs that encodes and automates specification revision in a trade-off for higher probability. The method uses co-safe LTL as the specification language and quantifies the revisions to the specification according to user-defined proposition costs. The framework computes a control policy that optimizes the trade-off between the probability of satisfaction and the cost of specification revision. The key idea of the method is a rule for the composition of the MDP, the automaton representing the specification, and the proposition costs such that all possible specification revisions along with their costs and probabilities of satisfaction are captured in one structure. The problem is then reduced to multi-objective optimization on an MDP. The power of the method is illustrated through simulations of a complex robotic scenario.

## I. INTRODUCTION

In recent years, there has been an increasing interest in automatic control generation for dynamical systems from high-level specifications (e.g., [1]–[7]). The main motivations for such studies are to eliminate human error and to lift the role of humans to the highest level of decision making – specification – by algorithmically constructing provably-correct control strategies. These approaches have specifically gained popularity in the fields of robotics, smart buildings, and systems biology. By the nature of these fields, the underlying systems include uncertainty and stochasticity, and hence the goal is to find an optimal strategy that maximizes the probability of satisfying the specification.

One major challenge that arises in these studies is the question of what to do if the optimal probability of satisfaction is not *good enough*? For example, consider the robotic scenario depicted in Fig. 1, where a home assistive robot carrying a tray of dishes is tasked to take the dishes to the kitchen without breaking them and without passing through the bedroom. Due to the object that is partially blocking the entrance to the kitchen from the common room, and robot’s sensor and actuation noise, the chances of entering the kitchen safely are less than ideal. Now, imagine that getting the dishes to the kitchen safely is of higher priority to the user than avoiding the bedroom. With this information,

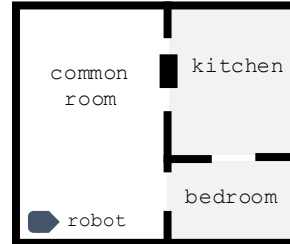


Fig. 1: A home assistive robot carrying a tray of dishes in the common room with the task of, “take the dishes to the kitchen by avoiding the bedroom”

then a possible desired behavior for the robot is to deliver the dishes to the kitchen through the bedroom given that the trade-off between the gained probability of success and the violation of the bedroom constraint is acceptable to the user. Such scenarios are commonly faced in robotics, and, more generally, in control of probabilistic systems in real world applications. In this work, we consider this problem from the specification perspective and investigate the following question: “how to automatically revise the specification so that the resulting strategy gives rise to an optimal trade-off between the probability of success and the degree of violation to the original specification.”

Many frameworks have been developed for control strategy synthesis from high-level specifications for stochastic systems [4], [5], [8]–[12]. These works typically use Markov decision processes (MDPs) as the (abstraction) model for the underlying systems and employ temporal logics, namely *linear temporal logic* (LTL) [13], as the specification language. In these works, the LTL specifications are viewed as hard constraints. Therefore, the developed frameworks are only able to compute a strategy that satisfies the specification optimally. In other words, by design, they are unable to suggest ways to improve the probability of success even if this probability is unsatisfactory.

In recent works, control generation methods that treat LTL specifications as soft constraints have been introduced [14]–[19]. These approaches are also referred to as *partial satisfaction* or *specification revision* methods, and their purpose is to enable (deterministic) robot motion planning in environments, in which the given high-level task is not fully satisfiable. The framework introduced in [14] allows for a temporary violation of the specification for simple transition systems. That method decomposes the specification into fragments and asks the user to prioritize them to generate a desirable plan. The works in [16], [17] also tackle the problem of

This work was supported by ERC Advanced Investigators Grant VERIWARE and EPSRC Mobile Autonomy Program Grant EP/M019918/1.

The authors are with the Dept. of Computer Science, University of Oxford, UK, E-mail: {morteza.lahijanian, marta.kwiatkowska}@cs.ox.ac.uk.

M. Lahijanian is the corresponding author.

planning for unsatisfiable LTL specifications for a transition system. These works propose several measures of distance between Büchi automata representing the specification and formulate (NP-hard) planning algorithms based on these measures. All of these works assume that the underlying system is deterministic, and their extensions to probabilistic systems are not clear and remain to be investigated.

The LTL planning frameworks in [18], [19] introduce three heuristics based on the notion of *graph distance* to generate the “closest” plan for deterministic robotics systems. One of these measures is adopted by [20] for control policy generation for MDPs with partially satisfiable LTL specifications. As discussed in [19], the used measure, however, is suitable for only a narrow fragment of specifications, namely coverage, e.g., eventually visit regions  $A$ ,  $B$ , and  $C$  in any order. For other types of specifications, this method of partial satisfaction generates undesirable behaviors.

Planning with soft constraints has also been studied in the AI community under the notion of *preference-based planning*, e.g., [21]–[23]. These works introduce various methods of reasoning about conditional preferences such as CP-nets [21]. The most related work from this community is [23], which introduces a probabilistic preference planning framework for MDPs. The preferences are expressed in LTL-style logic. The method reduces the problem to a quadratic programming problem and solves it by an SMT-solver. The proposed approach, however, is computationally expensive and belongs to the complexity class of NP-hard problems.

In previous work [24], a quantitative approach to LTL control generation for deterministic dynamical systems is introduced. The approach is based on the idea of *skipping* of the edges of the automaton that represents the specification. This idea leads to the notion of *distance to satisfaction* over the behaviors of the system and is derived from user-defined costs over atomic propositions. The method then automatically computes a series of controls (a plan) that minimizes the distance to satisfaction of the specification. This method is generally suitable for all types of specifications.

In this work, we propose an extension of the approach in [24] to probabilistic systems in the context of specification revision. Given an MDP, a specification in a fragment of LTL, and costs over propositions, the proposed scheme generates a control strategy that optimizes the trade-off between probability of satisfaction and the expected distance to satisfaction (degree of violation) to the original specification by considering all possible (allowed) revisions of the specification. This is enabled by a slight modification of the idea of *skipping* in [24] to *quantitative substitution* and a novel composition rule. Technically, the method first constructs a weighted automaton from the propositional costs and the specification automaton similar to [24]. The weighted automaton is then composed with the MDP, resulting in a product MDP. The composition rule is carefully designed so that the composed MDP (1) captures all possible specification revisions and (2) encodes both the distances of the revisions from the original specification and the probabilities of satisfaction. Finally, by using multi-objective optimization on the composed MDP, the

Pareto curve that captures all the optimal trade-offs between distance to satisfaction and probability of satisfaction and their corresponding strategies are computed.

Summarizing, the main contribution of this paper is the first LTL control synthesis framework for probabilistic systems that encodes and automates specification revision (partial satisfaction) capabilities suitable for all types of specifications, to the best of our knowledge. The framework is founded on a novel application of multi-objective verification [25], which is enabled by a new composition rule as discussed above. The power of the approach is illustrated through a robotic case study.

## II. PRELIMINARIES

### A. Markov Decision Processes

In this paper, we focus on Markov decision processes (MDPs) as models of probabilistic systems.

*Definition 1* (MDP): An MDP is a tuple  $\mathcal{M} = (Q, q_0, A, P, R, C, \Pi, L)$  where:

- $Q$  is a finite set of states;
- $q_0 \in Q$  is the initial state;
- $A$  is a set of actions;
- $P : Q \times A \rightarrow \text{Dist}(Q)$  is a transition probability function, mapping each state-action pair to a probability distribution over  $Q$  denoted by  $\text{Dist}(Q)$ ;
- $R : Q \times A \rightarrow \mathbb{R}^{\geq 0}$  is a reward function, assigning to each state-action pair a non-negative reward;
- $C : Q \times A \rightarrow \mathbb{R}^{\geq 0}$  is a cost function, assigning to each state-action pair a non-negative cost;
- $\Pi$  is a finite set of atomic propositions;
- $L : Q \rightarrow 2^\Pi$  is a labeling function that assigns to each state a set of atomic propositions from  $\Pi$ .

The set of available actions in  $q \in Q$  is denoted by  $A(q) \subseteq A$ . With an abuse of notation, we denote the transition probability from  $q$  to  $q'$  under action  $a \in A(q)$  by  $P(q, a, q')$ .

A *path* of an MDP  $\mathcal{M}$  is a possible sequence of states from the initial state through  $\mathcal{M}$ , i.e.,  $\gamma = \gamma_0 \xrightarrow{a_0} \gamma_1 \xrightarrow{a_1} \dots$ , where  $\gamma_0 = q_0$ ,  $\gamma_i \in Q$ ,  $a_i \in A(\gamma_i)$ , and  $P(\gamma_i, a_i, \gamma_{i+1}) > 0$  for all  $i \in \mathbb{N}$ . A path can be finite or infinite. We denote the set of all infinite and finite paths of  $\mathcal{M}$  by  $\text{Path}$  and  $\text{Path}^*$ , respectively. If  $\gamma \in \text{Path}^*$ ,  $\text{last}(\gamma)$  denotes the final state of  $\gamma$ . We define the *observation trace* of path  $\gamma$  to be  $w^\gamma = L(\gamma_0)L(\gamma_1)\dots$ .

*Example 1:* Consider the robotic scenario depicted in Fig. 1. An MDP  $\mathcal{M}_1$  representation of it is shown in Fig. 2. States  $q_0, q_1, q_2$  represent robot safely navigating in common room, kitchen, and bedroom, respectively, and  $q_3$  represents a crash that causes the dishes to break in any of the rooms. The actions and their corresponding transition probabilities, which indicate how the robot can move in the space, are shown by edges in this figure. A finite path of  $\mathcal{M}_1$  is  $q_0q_1$  with the observation trace of  $\{\text{common room}\}\{\text{kitchen}\}$ .

To reason about the behavior of the system represented by  $\mathcal{M}$ , we use *control policies* (also referred to as *strategies*, *adversaries*, or *schedulers*). A control policy specifies which action to choose at every state. In general, this choice can be history dependent and randomized.

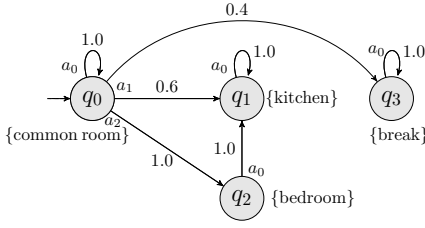


Fig. 2: An MDP  $\mathcal{M}_1$  representation of the robotic scenario depicted in Fig. 1

**Definition 2 (Control Policy):** A control policy of an MDP  $\mathcal{M}$  is a function  $\lambda : \text{Path}^* \rightarrow \text{Dist}(A)$  such that  $\lambda(\gamma, a) = 0$  for all  $a \notin A(\text{last}(\gamma))$ , where,  $\lambda(\gamma, a)$  is the probability of choosing  $a$  at state  $\text{last}(\gamma)$  under  $\lambda$ .

We denote the set of all policies of  $\mathcal{M}$  by  $\Lambda_{\mathcal{M}}$ . Policy  $\lambda \in \Lambda_{\mathcal{M}}$  is *deterministic* if  $\lambda(\gamma)$  is a point distribution for all  $\gamma \in \text{Path}^*$  (i.e.,  $\lambda$  chooses an action in  $A(\text{last}(\gamma))$  with probability 1); otherwise,  $\lambda$  is *randomized*. A policy  $\lambda$  is *memoryless (stationary)* if  $\lambda(\gamma)$  depends only on  $\text{last}(\gamma)$ .

### B. Syntactically Co-safe LTL

We use syntactically co-safe LTL [26] to write the specifications of the probabilistic system.

**Definition 3 (syntax):** Let  $\Pi = \{p_1, p_2, \dots, p_N\}$  be a set of Boolean atomic propositions. A syntactically co-safe LTL formula over  $\Pi$  is inductively defined as following:

$$\varphi := p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathcal{X}\varphi \mid \varphi \mathcal{U}\varphi \mid \mathcal{F}\varphi$$

where  $p \in \Pi$ ,  $\neg$  (negation),  $\vee$  (disjunction), and  $\wedge$  (conjunction) are Boolean operators, and  $\mathcal{X}$  (“next”),  $\mathcal{U}$  (“until”), and  $\mathcal{F}$  (“eventually”) are temporal operators.

**Definition 4 (Semantics):** The semantics of syntactically co-safe LTL formulas are defined over infinite traces over  $2^\Pi$ . Let  $w = \{w_i\}_{i=1}^\infty$  with  $w_i \in 2^\Pi$  be an infinite trace and  $w^i = w_i w_{i+1} \dots$  be the  $i$ -th suffix.  $w \models \varphi$  indicates that  $w$  satisfies formula  $\varphi$  and is recursively defined as following:

- $w \models p$  if  $p \in w_1$ ;
- $w \models \neg p$  if  $p \notin w_1$ ;
- $w \models \varphi_1 \vee \varphi_2$  if  $w \models \varphi_1$  or  $w \models \varphi_2$ ;
- $w \models \varphi_1 \wedge \varphi_2$  if  $w \models \varphi_1$  and  $w \models \varphi_2$ ;
- $w \models \mathcal{X}\varphi$  if  $w^1 \models \varphi$ ;
- $w \models \varphi_1 \mathcal{U} \varphi_2$  if  $\exists k \geq 0, w^k \models \varphi_2$ , &  $\forall i \in [0, k), w^i \models \varphi_1$ ;
- $w \models \mathcal{F}\varphi$  if  $\exists k \geq 0, w^k \models \varphi$ .

It is important to note that finite traces are sufficient to satisfy syntactically co-safe LTL formulas, even though these formulas have infinite-time semantics. We say that a path of an MDP satisfies  $\varphi$ , if its observation trace satisfies  $\varphi$ .

Given a co-safe LTL formula  $\varphi$ , a *deterministic finite automaton* (DFA) that precisely accepts all the finite traces that satisfy  $\varphi$  can be constructed [26].

**Definition 5 (DFA):** A DFA is given by a tuple  $\mathcal{A} = (Z, \Sigma, \delta, z_0, F)$ , where  $Z$  is a finite set of states,  $\Sigma$  is the input alphabet,  $\delta : Z \times \Sigma \rightarrow Z$  is the transition function,  $z_0 \in Z$  is the initial state, and  $F \subseteq Z$  is the set of accepting states. The transition function  $\delta$  can be also viewed as a relation  $\delta \subseteq Z \times \Sigma \times Z$ , where every transition is a tuple  $(z_1, \sigma, z_2) \in \delta$  iff  $z_2 = \delta(z_1, \sigma)$ . A finite *run* of  $\mathcal{A}$  on a trace

$w = w_1 \dots w_n$  is a sequence of states  $\mu = \mu_0 \mu_1 \dots \mu_n$ , where  $\mu_0 = z_0$ ,  $\mu_i \in Z$ , and  $(\mu_{i-1}, w_i, \mu_i) \in \delta$  for  $i = 1, \dots, n$ .  $\mu$  is called an *accepting run* if  $\mu_n \in F$ .

We denote the DFA that is constructed from a formula  $\varphi$  by  $\mathcal{A}_\varphi$ . An input trace  $w$  that induces an accepting run  $\mu$  in  $\mathcal{A}_\varphi$  is called  $\varphi$ -satisfying. To reason quantitatively over the satisfaction of  $\varphi$ , we employ *weighted DFA* (WDFA).

**Definition 6 (WDFA):** A *weighted DFA* (WDFA) is a tuple  $\mathcal{A}^\rho = (\mathcal{A}, \rho)$ , where  $\mathcal{A}$  is a DFA, and  $\rho : \delta \rightarrow \mathbb{R}$  assigns a weight for every transition in  $\delta$ . Consider a trace  $w = w_1 \dots w_n$ , and let  $\mu = \mu_0 \mu_1 \dots \mu_n$  be the run of  $\mathcal{A}^\rho$  on  $w$ , i.e.,  $(\mu_{i-1}, w_i, \mu_i) \in \delta$  for all  $i \in \{1, \dots, n\}$ . We define the weight of  $w$ , with an abuse of notation, to be  $\rho(w) = \sum_{i=1}^n \rho(\mu_{i-1}, w_i, \mu_i)$ . Thus, the weight of a trace is the sum of weights along the run of  $\mathcal{A}$  on it. Therefore, a WDFA defines a function  $h_{\mathcal{A}^\rho} : \Sigma^* \rightarrow \mathbb{R}$ .

## III. PROBLEM FORMULATION

The encompassing goal of this work is a control policy synthesis scheme for probabilistic systems that enables the increase of the probability of satisfaction of a high-level specification at the cost of revision (violation) of the specification. In other words, we aim to compute the control policies that give rise to the optimal trade-offs between probabilities of satisfaction of the revised specifications and the degrees of violation to the original specification.

We assume that the probabilistic system is in the form of an MDP, and the specification is given as a co-safe LTL formula  $\varphi$  over the MDP’s set of atomic propositions  $\Pi = \{p_1, \dots, p_N\}$ . Furthermore, each ordered pair of propositions  $(p_i, p_j)$  is associated with a non-negative cost, representing the cost of violation (substitution) of  $p_j$  by  $p_i$ . Let  $c : \Pi \times \Pi \rightarrow \mathbb{R}^{\geq 0}$ , where  $c(p_i, p_i) = 0$ , denote this (possibly partial) cost function. The intuition is that the user allows revisions of the specification  $\varphi$  through substitution of its propositions, and  $c(p_i, p_j)$  captures the degree of user’s willingness on the substitution of  $p_j$  by  $p_i$ .

Instead of actually revising the specification, we evaluate these costs over the paths of the MDP. We employ  $c$  to measure how far a path is from satisfying  $\varphi$  through allowed substitutions in its observation trace. Informally, we define “distance to satisfaction” of a path to  $\varphi$  to be the total cost of such substitutions (formally defined in Sec. IV). Since the paths of the MDP are probabilistic, we reason about their distances to satisfaction in the form of expectation.

**Example 2:** To illustrate this setting, consider the robotic scenario depicted in Fig. 1 and its MDP representation  $\mathcal{M}_1$  in Fig. 2. The specification is  $\varphi_1 = ((\neg \text{break} \wedge \neg \text{bedroom}) \mathcal{U} (\neg \text{break} \wedge \text{kitchen}))$ . The (simplified) DFA  $\mathcal{A}_{\varphi_1}$  is shown in Fig. 3. Clearly, the robot can satisfy  $\varphi$  only by going straight to the kitchen from the common room. This path corresponds to the MDP path  $q_0 q_1$  and the trace of  $\{\text{common room}\} \{\text{kitchen}\}$  with the probability of 0.6. To achieve this path, the robot has to choose action  $a_1$  in state  $q_0$ , which also could result in breaking the dishes with the probability of 0.4. Since the user considers keeping the dishes safe of much higher priority than going

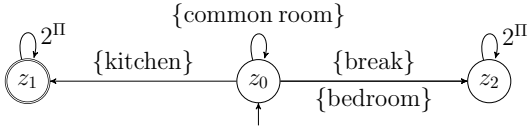


Fig. 3: A (simplified) DFA  $\mathcal{A}_{\varphi_1}$  for  $\varphi_1 = ((\neg \text{break} \wedge \neg \text{bedroom}) \mathcal{U} (\neg \text{break} \wedge \text{kitchen}))$

to the bedroom or never reaching the kitchen, she assigns the cost of 1 to the violation of common room by bedroom (pretending the bedroom is the common room), i.e.,  $c_1(\text{bedroom}, \text{common room}) = 1$ , the cost of 10 for the violation of kitchen by common room or bedroom (pretending the common room or the bedroom is the kitchen), i.e.,  $c_1(\text{common room}, \text{kitchen}) = c_1(\text{bedroom}, \text{kitchen}) = 10$ , and extremely large costs to all the other violations to prevent them from happening. Now, the robot has several ways to get to the kitchen. For instance, it can take the path through the bedroom and endure the cost of 1 in trade-off for the probability of satisfaction of 1. It could also stay in the common room and never break the dishes at the cost of 10 for violating kitchen by common room. Given all the trade-offs, then the user can choose the desired behavior, which is the former one in this case.

Therefore, we are interested in computing a policy for the MDP with two objectives: maximizing the probability of satisfaction and minimizing the expected distance to satisfaction of  $\varphi$  according to the user's preference. The formal statement of the problem is as follows.

**Problem 1:** Given an MDP  $\mathcal{M}$ , a co-safe LTL formula  $\varphi$ , and a substitution cost function  $c$  over pairs of atomic propositions, synthesize a control policy that maximizes the probability of satisfaction of  $\varphi$  and minimizes the expected distance to satisfaction of  $\varphi$ .

To approach Problem 1, we first generate a WDFA from DFA  $\mathcal{A}_\varphi$  and cost function  $c$  to capture all possible violations (revisions) of  $\varphi$  with their corresponding distances to satisfaction. Then, we compose this WDFA with the MDP through a specific composition rule that translates the distance to satisfaction that is defined over traces to the paths of the product MDP. At the same time, the probability of satisfaction of  $\varphi$  is also captured by the product MDP through a reward assignment by the composition rule. Therefore, the problem is reduced to a two-objective optimization problem over the product MDP. This optimization problem can be solved by either methods of *value iteration* (VI) or *linear programming* (LP) in polynomial time [25].

#### IV. SYNTHESIS FRAMEWORK

In this section, we introduce our control policy synthesis framework. We note that, for completeness, we first review the definition of distance to satisfaction from [24] and then introduce the methodology, which is the main contribution of the paper, including a new semantics to lift the proposition substitution costs given by  $c$  to the alphabets of  $\mathcal{A}_\varphi$ .

##### A. Construction of WDFA

Here, we detail the construction of a WDFA from  $\varphi$  and  $c$  to capture the violation costs between two traces. This WDFA is utilized to measure the distance to satisfaction of the observation traces of MDP  $\mathcal{M}$  to  $\varphi$ . Recall that, from a co-safe LTL formula  $\varphi$ , a DFA  $\mathcal{A}_\varphi$  that accepts precisely all satisfying traces of  $\varphi$  can be constructed. The formula  $\varphi$  is defined over the propositions in  $\Pi$ , and hence the alphabets (also known as letters) of  $\mathcal{A}_\varphi$  are from  $2^\Pi$ , i.e.,  $\Sigma = 2^\Pi$ . Each state of the MDP  $\mathcal{M}$  is also labeled with an element of  $2^\Pi$ . Therefore, the finite traces from both  $\mathcal{A}_\varphi$  and  $\mathcal{M}$  are in  $(2^\Pi)^*$ . To distinguish between them, we denote a trace of (a path of)  $\mathcal{M}$  by  $w^\mathcal{M}$  and a trace of  $\mathcal{A}_\varphi$  by  $w^\varphi$ . The set of all traces of  $\mathcal{M}$  is called the *language* of  $\mathcal{M}$  and is denoted by  $\mathcal{L}(\mathcal{M})$ . Similarly, the language of  $\varphi$  is the set of all accepting traces of  $\mathcal{A}_\varphi$  and is denoted by  $\mathcal{L}(\varphi)$ .

To define the distance to satisfaction of  $w^\mathcal{M}$  to  $\varphi$ , we first need to measure the substitution cost of a letter  $\sigma^\mathcal{M}$  in  $w^\mathcal{M}$  for another one  $\sigma^\varphi$  in  $w^\varphi$ . Recall that each letter is a set of propositions. Therefore, we can use  $c$ , the propositional substitution cost function, to derive the cost of letter substitutions. One can think of several ways to do this. For instance, the work in [24] proposes two semantics for it. These semantics, however, are specific to the case that the violation costs of an atomic proposition by all other ones are equal. In this work, we give more freedom to the user by allowing different costs for the violation of a proposition (depending on the substitution proposition). Therefore, we suggest the following semantics.

Let  $\eta : 2^\Pi \times 2^\Pi \rightarrow \mathbb{R}$  be the letter substitution cost function, where  $\eta(\sigma^\mathcal{M}, \sigma^\varphi)$  returns the cost of substituting  $\sigma^\mathcal{M}$  for  $\sigma^\varphi$ . Also, let  $l = \max(|\sigma^\mathcal{M}|, |\sigma^\varphi|)$ , where  $|\sigma|$  is the number of propositions in  $\sigma$ . Then,  $\eta(\sigma^\mathcal{M}, \sigma^\varphi) = \min_{(\alpha_i, \beta_i) \in (\sigma^\mathcal{M} \times \sigma^\varphi)} \sum_{i=1}^l c(\alpha_i, \beta_i)$  such that  $\bigcup_{i=1}^l \alpha_i = \sigma^\mathcal{M}$  and  $\bigcup_{i=1}^l \beta_i = \sigma^\varphi$ . Using  $\eta$ , we define the distance between two traces  $w^\mathcal{M}$  and  $w^\varphi$  as follows.

$$\text{DIST}(w^\mathcal{M}, w^\varphi) = \begin{cases} \sum_{i=1}^{|w^\varphi|} \eta(w_i^\mathcal{M}, w_i^\varphi) & \text{if } |w^\mathcal{M}| = |w^\varphi| \\ \infty & \text{otherwise.} \end{cases}$$

We formally define distance to satisfaction of  $w^\mathcal{M}$  to  $\varphi$  as:

$$\text{DISTOSAT}(w^\mathcal{M}, \varphi) = \min_{w^\varphi \in \mathcal{L}(\varphi)} \text{DIST}(w^\mathcal{M}, w^\varphi).$$

In words,  $\text{DISTOSAT}(w^\mathcal{M}, \varphi)$  is the minimum sum of the costs of the letter substitutions required to turn  $w^\mathcal{M}$  to an accepting trace of  $\varphi$ .

**Example 3:** Consider again the robotic scenario depicted in Fig. 1 and the robot behavior of taking the dishes to the kitchen through the bedroom. The MDP trace of this behavior is  $w^\mathcal{M} = \{\text{common room}\}\{\text{bedroom}\}\{\text{kitchen}\}$ . The closest accepting trace of  $\mathcal{A}_\varphi$  (see Fig. 3) to  $w^\mathcal{M}$  is  $w^\varphi = \{\text{common room}\}\{\text{common room}\}\{\text{kitchen}\}$ . It is clear that the robot violates (substitutes) letter  $w_2^\varphi = \{\text{common room}\}$  by  $w_2^\mathcal{M} = \{\text{bedroom}\}$ , whose cost is 1. Therefore,  $\text{DISTOSAT}(w^\mathcal{M}, \varphi) = 1$  for this robot behavior. Another possible behavior is for the robot to remain in the common room, resulting in the trace  $w^\mathcal{M} =$

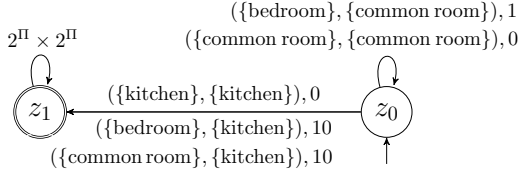


Fig. 4: W DFA  $\mathcal{A}_{\varphi_1}^{\rho}$  constructed from DFA  $\mathcal{A}_{\varphi_1}$  (Fig. 3) and proposition substitution cost function  $c_1$  in Example 2.

$\{\text{common room}\}\{\text{common room}\}$ . The closest accepting trace to it is  $w^{\varphi} = \{\text{common room}\}\{\text{kitchen}\}$ . Here,  $w_2^{\mathcal{M}} = \{\text{common room}\}$  is substituted for  $w_2^{\varphi} = \{\text{kitchen}\}$ . The cost of this substitution is 10; hence,  $\text{DISTOSAT}(w^{\mathcal{M}}, \varphi) = 10$ . Obviously, the former behavior, which has a smaller distance to satisfaction, is preferred by the user.

To enable algorithmic computation of distance to satisfaction, we construct W DFA  $\mathcal{A}_{\varphi}^{\rho}$  from  $\mathcal{A}_{\varphi} = (Z, \Sigma, \delta, z_0, F)$ , where  $\Sigma = 2^{\Pi}$ . We define the alphabets of  $\mathcal{A}_{\varphi}^{\rho}$  to be the product of the alphabets of  $\mathcal{A}_{\varphi}$  to allow substitution. The cost of the substitutions is then given by the weight function  $\rho$ . Formally,  $\mathcal{A}_{\varphi}^{\rho} = (Z, \Sigma \times \Sigma, \delta^{\rho}, z_0, F, \rho)$ , where for every  $z \in Z$  and letter  $(\sigma^{\mathcal{M}}, \sigma^{\varphi}) \in \Sigma \times \Sigma$ , the transition  $\delta^{\rho}(z, (\sigma^{\mathcal{M}}, \sigma^{\varphi})) = \delta(z, \sigma^{\varphi})$ , and the weight of the transition is given by  $\rho(z, (\sigma^{\mathcal{M}}, \sigma^{\varphi}), z') = \eta(\sigma^{\mathcal{M}}, \sigma^{\varphi})$ . Note that, by design, the accepting input trace  $w^{\varphi} \in \Sigma^*$  in  $\mathcal{A}_{\varphi}$  results in the same accepting run as the input trace of  $(w^{\mathcal{M}}, w^{\varphi}) \in (\Sigma \times \Sigma)^*$  in  $\mathcal{A}_{\varphi}^{\rho}$ , independent of  $w^{\mathcal{M}}$ . The total weight of this run in  $\mathcal{A}_{\varphi}^{\rho}$ , however, depends on  $w^{\mathcal{M}}$  and is, in fact, the distance between  $w^{\mathcal{M}}$  and  $w^{\varphi}$ .

The construction of the W DFA is not computationally expensive. The state space of the W DFA is the same as the DFA, and its number of edges is polynomial (quadratic in the worst case) in the number of edges of the DFA. For implementation purposes, it is not necessary to construct the W DFA. It is preferred to simulate it on the fly by DFA and  $\eta$ . Lastly, we suggest the use of a minimized DFA to eliminate unnecessary computation or simulation of the W DFA.

*Example 4:* Consider the task DFA  $\mathcal{A}_{\varphi_1}$  shown in Fig. 3 and the proposition substitution costs  $c_1$  in Example 2. In the first step, we minimize  $\mathcal{A}_{\varphi_1}$  to reduce unnecessary computations. The minimized DFA contains only  $z_0, z_1$ , and their corresponding edges and labels. Then W DFA  $\mathcal{A}_{\varphi_1}^{\rho}$  is constructed by pairing each existing label with a letter that can substitute it as shown in Fig. 4. The cost of this substitution, then, becomes the weight of the edge enabled by this newly generated label.

### B. Product MDP

To generate a control policy for the MDP  $\mathcal{M}$  that optimizes the trade-off between the maximum probability of satisfaction of  $\varphi$  and its minimum expected distance to satisfaction, we compose MDP  $\mathcal{M}$  with  $\mathcal{A}_{\varphi}^{\rho}$ , i.e.,  $\mathcal{M}^{\mathcal{P}} = \mathcal{M} \times \mathcal{A}_{\varphi}^{\rho}$ . The resulting structure is an MDP  $\mathcal{M}^{\mathcal{P}} = (Q^{\mathcal{P}}, q_0^{\mathcal{P}}, A^{\mathcal{P}}, P^{\mathcal{P}}, R^{\mathcal{P}}, C^{\mathcal{P}})$ , where

- $Q^{\mathcal{P}} = (Q \times Z) \cup \{q_g^{\mathcal{P}}, q_b^{\mathcal{P}}\}$  is a set of product states, where  $q_g^{\mathcal{P}}$  and  $q_b^{\mathcal{P}}$  are the “good” and “bad” terminal

(sink) states;

- $q_0^{\mathcal{P}} = (q_0, z_0)$  is the initial state;
- $A^{\mathcal{P}} = \{A^{\mathcal{P}}(q^{\mathcal{P}}) \mid q^{\mathcal{P}} \in Q^{\mathcal{P}}\}$  is the set of actions;
- $P^{\mathcal{P}} : Q^{\mathcal{P}} \times A^{\mathcal{P}} \rightarrow \text{Dist}(Q^{\mathcal{P}})$  is a probability distribution assigning to each transition  $(q^{\mathcal{P}}, a^{\mathcal{P}}, q^{\mathcal{P}'})$  a probability in accordance to  $P$ ;
- $R^{\mathcal{P}} : Q^{\mathcal{P}} \times A^{\mathcal{P}} \rightarrow \mathbb{R}^{\geq 0}$  is a reward function designed to capture the probability of satisfying  $\varphi$  as explained below;
- $C^{\mathcal{P}} : Q^{\mathcal{P}} \times A^{\mathcal{P}} \rightarrow \mathbb{R}^{\geq 0}$  is a cost function designed to capture the distance to satisfaction of the paths of  $\mathcal{M}$  as explained below.

For simplicity of presentation, we explain the generation of  $\mathcal{M}^{\mathcal{P}}$  in two steps. In the first step, we focus only on the subspace  $(Q \times Z)$  of  $\mathcal{M}^{\mathcal{P}}$ . In the second step, we complete the construction of the product by considering the terminal states  $q_g^{\mathcal{P}}$  and  $q_b^{\mathcal{P}}$ .

1) *Pre-Processing Step:* A key step to the construction of  $\mathcal{M}^{\mathcal{P}}$  is the generation of the set of actions  $A^{\mathcal{P}}$  from the set of actions  $A$  in  $\mathcal{M}$ . The goal is to capture all possible substitutions of the labels of the states of  $\mathcal{M}$  in the structure of  $\mathcal{M}^{\mathcal{P}}$ . We do this by a unique construction of  $A^{\mathcal{P}}$ . The intuition behind it is as follows. Under action  $a \in A(q)$ , state  $q \in Q$  can possibly have several successors with non-zero probabilities. The label of each successor state, in our framework, can be substituted by another label at a certain cost. Therefore, under  $a$ , several combinations of successor labels are possible. For each successor label combination under  $a$ , we define an action in  $A^{\mathcal{P}}$  (a copy of  $a$ ).

Formally, let  $q'_1, \dots, q'_k$  be all the successor states of the state-action pair  $(q, a)$  in  $\mathcal{M}$ , where  $q'_i \in Q$  and  $P(q, a, q'_i) > 0$ . Moreover, let  $\mathbf{L}(q) \subseteq 2^{\Pi}$  denote the set of all letters (labels) that can substitute the label of  $q$ . Then, we define the set of available actions at product state  $q^{\mathcal{P}} = (q, z)$  to be

$$A^{\mathcal{P}}((q, z)) = \left\{ (a, l(q'_1), \dots, l(q'_k)) \mid a \in A(q) \text{ and } l(q'_i) \in \mathbf{L}(q'_i) \forall i \in \{1, \dots, k\} \right\}.$$

Under action  $a^{\mathcal{P}} = (a, l(q'_1), \dots, l(q'_k))$  in  $A^{\mathcal{P}}((q, z))$ , the transition probability  $P^{\mathcal{P}}((q, z), a^{\mathcal{P}}, (q'_i, z')) =$

$$\begin{cases} P(q, a, q'_i) & \text{if } \delta^{\rho}(z, (l(q'_i), L(q'_i))) = z' \\ 0 & \text{otherwise.} \end{cases}$$

If  $\delta^{\rho}(z, (l(q'_i), L(q'_i))) = z'$ , we assign the cost of  $\bar{C}((q, z), a^{\mathcal{P}}, (q'_i, z')) = \eta(l(q'_i), L(q'_i))$  to this edge. We use these edge costs to compute the costs of the state-action pairs  $C^{\mathcal{P}}(q^{\mathcal{P}}, a^{\mathcal{P}})$  in the post-processing step.

2) *Post-Processing Step:* In this step, we complete the construction of  $\mathcal{M}^{\mathcal{P}}$  such that it can be directly utilized for a multi-objective optimization computation. We achieve this by adding two terminal states  $q_g^{\mathcal{P}}$  and  $q_b^{\mathcal{P}}$  to the states space of  $\mathcal{M}^{\mathcal{P}}$  and taking the necessary steps to turn them into sink states. These steps are as following:

- **Fixing  $P^{\mathcal{P}}$ :** if the sum of the transition probabilities of state-action pair  $((q, z), a^{\mathcal{P}})$ , where  $a^{\mathcal{P}} \in A^{\mathcal{P}}((q, z))$ , is less than 1, assign the remaining probability mass

- **Accepting States to  $q_g^P$ :** add an action to every state  $(q, z)$  with  $z \in F$  (accepting state), to enable a transition to  $q_g^P$  with probability 1, i.e., add  $a^P$  to  $A^P((q, z))$ , where  $z$  is accepting, with  $P^P((q, z), a^P, q_g^P) = 1$ .
- **Sink States:** remove all actions that cause a self-transition with probability 1 from the states in  $(Q \times Z)$ . Then, turn  $q_g^P$  and  $q_b^P$  into sink states by adding an action to each with a self-transition probability of 1.
- **Rewards:** assign a reward of 1 to the state-actions pairs that have a transition probability 1 to  $q_g^P$  and 0 to all the other ones. That is,  $R^P(q^P, a^P) = 1$  if  $q^P = (q, z)$  and  $z \in F$ ; otherwise,  $R^P(q^P, a^P) = 0$ .
- **Costs:** assign the cost of 0 to all the edges that do not have a cost assignment already. Then, the state-action pair costs become the expectation of the edge costs  $\bar{C}$ , i.e.,  $C^P(q^P, a^P) = \sum_{q^{P'} \in Q^P} P^P(q^P, a^P, q^{P'}) \bar{C}(q^P, a^P, q^{P'})$ .

The complexity of the construction of the product MDP  $\mathcal{M}^{\mathcal{P}}$  is exponential in the number of allowed substitutions. This is the classic combinatorial problem because all possible combinations of the allowed substitutions need to be considered. In the worst case, every letter in  $\Sigma$  can substitute every other letter, and each state-action pair in  $\mathcal{M}$  has a transition to all the states in  $Q$  with non-zero probability. Then, the number of actions at each state of  $\mathcal{M}^{\mathcal{P}}$  is  $|\mathcal{A}^{\mathcal{P}}((q, z))| = |A(q)| \cdot |\Sigma|^{|Q|}$ . Hence, the size of the product MDP in the worst case is  $|\mathcal{M}^{\mathcal{P}}| = |Z| \cdot |\Sigma|^{|Q|} \sum_{q \in Q} |A(q)|$ .

### C. Multi-Objective Optimization

By the construction of the product MDP, Problem 1 reduces to a two-objective optimization problem over  $\mathcal{M}^{\mathcal{P}}$ . Under a control policy  $\lambda \in \Lambda_{\mathcal{M}^{\mathcal{P}}}$ , the probability of satisfaction of  $\varphi$  by the paths of  $\mathcal{M}^{\mathcal{P}}$  (see [27]) is

$$\text{PROBSAT}_{\mathcal{M}^{\mathcal{P}}}^{\lambda}[\varphi] = \mathbb{E}_{\mathcal{M}^{\mathcal{P}}}^{\lambda} \left( \sum_{k=0}^{\infty} R^{\mathcal{P}}(q_k^{\mathcal{P}}, \lambda(q_k^{\mathcal{P}})) \right), \quad (1)$$

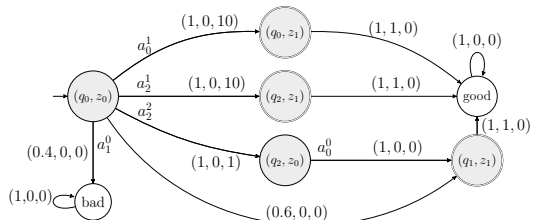


Fig. 5: Product MDP  $\mathcal{M}_1^{\mathcal{P}} = \mathcal{M}_1 \times \mathcal{A}_{\varphi_1}^{\rho}$ .

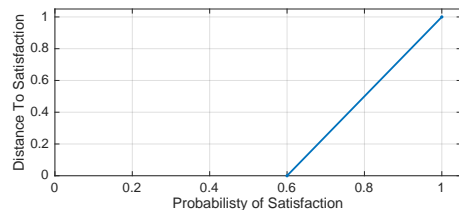


Fig. 6: The Pareto curve computed on  $\mathcal{M}_1^P$  (Fig. 5) for the robotic scenario in Fig. 1.

where  $\mathbb{E}_{\mathcal{M}^{\mathcal{P}}}^{\lambda}$  denotes the expectation operator over the paths of  $\mathcal{M}^{\mathcal{P}}$  under  $\lambda$ . Similarly, the expected distance to satisfaction of the paths of  $\mathcal{M}^{\mathcal{P}}$  to  $\varphi$  under  $\lambda$  is

$$\text{DisToSat}_{\mathcal{M}^P}^\lambda[\varphi] = \mathbb{E}_{\mathcal{M}^P}^\lambda \left( \sum_{k=0}^{\infty} C^P(q_k^P, \lambda(q_k^P)) \right). \quad (2)$$

Therefore, the multi-objective problem becomes the computation of a control policy  $\lambda \in \Lambda_{\mathcal{MP}}$  that maximizes (1) and minimizes (2). The synthesis of this control policy can be achieved by solving an LP problem [25].

In this work, we first compute the Pareto curve [28] to illustrate all the optimal trade-offs between the two objectives by using VI to solve the problem. Then, by the user’s choice of a point on the curve, the corresponding control policy can be obtained by an LP solver. The obtained control policy is memoryless and generally randomized for MDP  $\mathcal{M}^P$ . Through a simple projection, it can be converted to a finite-memory (history dependent) control policy for the MDP  $\mathcal{M}$ .

*Example 6:* The Pareto curve obtained for the robotic scenario in Fig. 1 computed on the product MDP  $\mathcal{M}_1^P$  (Fig. 5) is shown in Fig. 6. As the curve illustrates, delivering the dishes safely to the kitchen comes at the trade-off of violating the bedroom constraint at the cost of 1, whereas this probability reduces to 0.6 if no violation is allowed.

We note that, at a cost of additional complexity, the proposed framework can be modified in a straightforward manner to enable the substitution of single letters by a sequence of letters (finite trace). This flexibility might be interesting from the application point of view. For instance, a robot can substitute an important task only by performing a sequence of other tasks.

## V. CASE STUDY

We evaluated the performance of the proposed control policy synthesis with specification revision framework on a robotic scenario in an indoor environment in simulations. For two different task specifications, each paired with an

atomic substitution cost function, we showed the range of possible probabilities of satisfaction and their corresponding distances to satisfaction. In other words, we showed how the maximum probability of satisfaction can be improved through specification revision.

We considered the environment shown in Fig. 9d, which consists of 4 rooms, obstacles, and 5 regions of interest. The initial position of the robot is shown in blue. The motion of the robot in this environment was assumed to be given as an MDP obtained through a discretization of the environment to a  $10 \times 10$  grid. Each grid cell was associated to a state of the MDP. Moreover, the bounding walls of the environment were represented by a single state of the MDP. Hence, the robot MDP had a total of 101 states.

The set of atomic propositions were  $\Pi = \{Obs, p_0, \dots, p_5\}$ , and the labeling of the MDP states was done according to the environment map (see Fig. 9d), where  $p_0$  is the label of all the white cells. The state representing the bounding walls was labeled with  $\{Obs\}$ . The states corresponding to the cells containing (black) walls were also labeled with  $\{Obs\}$ . Furthermore, there were 5 actions available to the robot in each MDP state:  $A = \{a_{north}, a_{east}, a_{south}, a_{west}, a_{terminate}\}$ . The outcome of  $a_{terminate}$  was a deterministic termination in the current cell of the robot. The transition probability distribution of the rest of the actions for any cell is shown in Fig. 7.

The first robot specification was, “Visit regions  $p_2$ ,  $p_4$ , and  $p_1$ , in that order and always avoid obstacles.” This specification translates to the following co-safe LTL formula:

$$\varphi_2 = \neg Obs \mathcal{U} \left( p_2 \wedge (\neg Obs \mathcal{U} (p_4 \wedge (\neg Obs \mathcal{U} p_1))) \right).$$

The atomic proposition substitution costs for this specification were:  $c_2(p_0, p_2) = 1$ ,  $c_2(p_0, p_4) = 2$ ,  $c_2(p_0, p_1) = 3$ , and  $\infty$  for all the others. These costs indicate the importance of visiting  $p_1$ ,  $p_2$ , and  $p_4$  in specification  $\varphi_2$ , and user’s willingness to allow them to be ignored by the robot.

The second specification was, “Go to  $p_5$  without passing through  $p_2$  and colliding with obstacles,” which translates to the co-safe LTL formula  $\varphi_3 = (\neg p_2 \wedge \neg Obs) \mathcal{U} p_5$ . For this specification, the user was willing to compromise  $p_5$  by allowing the visit of  $p_4$ ,  $p_1$ , or  $p_3$  instead to achieve a higher probability of satisfaction, according to the costs  $c_3(p_4, p_5) = 1$ ,  $c_3(p_1, p_5) = 5$ , and  $c_3(p_3, p_5) = 10$ . The user was also willing to ignore  $p_5$  at the cost  $c_3(p_0, p_5) = 20$ .

We used the proposed framework to compute the Pareto curve for each of the specifications (see Fig. 8). We also generated 50 sample paths under four control policies, each corresponding to a Pareto point. They are shown in Fig. 9. For  $\varphi_2$ , the probability of satisfaction with no violation was 0.36, i.e., distance to satisfaction of 0 (see Fig. 9a). The bottleneck of this task for the robot was to visit  $p_4$ . By violating this region, the robot could complete  $\varphi_2$  with the much higher probability of 0.88 (Fig. 9b). The distance to satisfaction of this behavior was 1.9. At the distance to satisfaction of 4.0, the robot could complete the task with the probability of 0.99. The robot behavior under this control

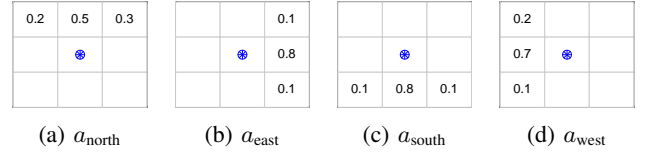


Fig. 7: Transition probability distribution under each action.

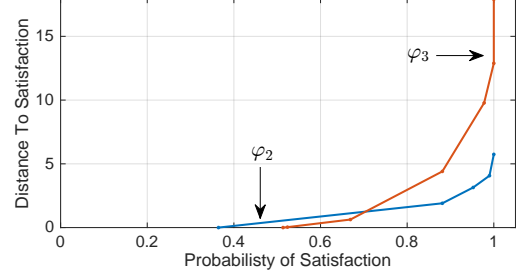


Fig. 8: Pareto curves for  $\varphi_2$  and  $\varphi_3$

policy was interesting because it chose to terminate when the robot found itself in a location that had a high probability of colliding with an obstacle as shown in Fig. 9c. Lastly, at distance to satisfaction of 5.7, the robot substituted  $p_0$  for all the specified regions to achieve probability of 1 by remaining in its initial position (Fig. 9d).

The probability of satisfaction of  $\varphi_3$  with no violation was 0.52 (Fig. 9e). By allowing the substitution of  $p_4$  for  $p_5$ , the probability of satisfaction went up to 0.67. As shown in Fig. 9f, the robot chose between  $p_5$  and  $p_4$  according to its probability of success from the current region. For the distance to satisfaction of 4.4 and 9.8, the robot could increase its probability of success to 0.88 and 0.98, respectively, by going to  $p_1$  and  $p_3$  instead of  $p_5$  (Figs. 9g and 9h). We verified all of these probabilities by 1,000 simulations. The results were within 2% of the theoretical values.

The size of the product MDPs (sum of all state-action pairs) for  $\varphi_2$  and  $\varphi_3$  were 3,257 and 3,930, respectively. Their constructions in MATLAB took 13.0s and 21.3s, respectively. We used PRISM [29] to compute the Pareto curves and control policies, each of which took less than 2s. All these computations were performed on a MACBOOK PRO with a 2.7 GHz processor and 8 GB of RAM.

## VI. CONCLUSION

In this paper, we introduced a computational framework for specification revision for MDPs. The framework generates control policies by optimizing the trade-off between the probability of satisfaction of a high-level specification and its revision cost. We focused on co-safe LTL as the specification language and used user-defined substitution costs over the atomic propositions to define quantitative revisions of the specification. The efficacy of the method was demonstrated through simulations of a robotic scenario.

## REFERENCES

- [1] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, “Where’s waldo? sensor-based temporal logic motion planning,” in *Int. Conf. on Rob. and Aut.*, 2007, pp. 3116–3121.

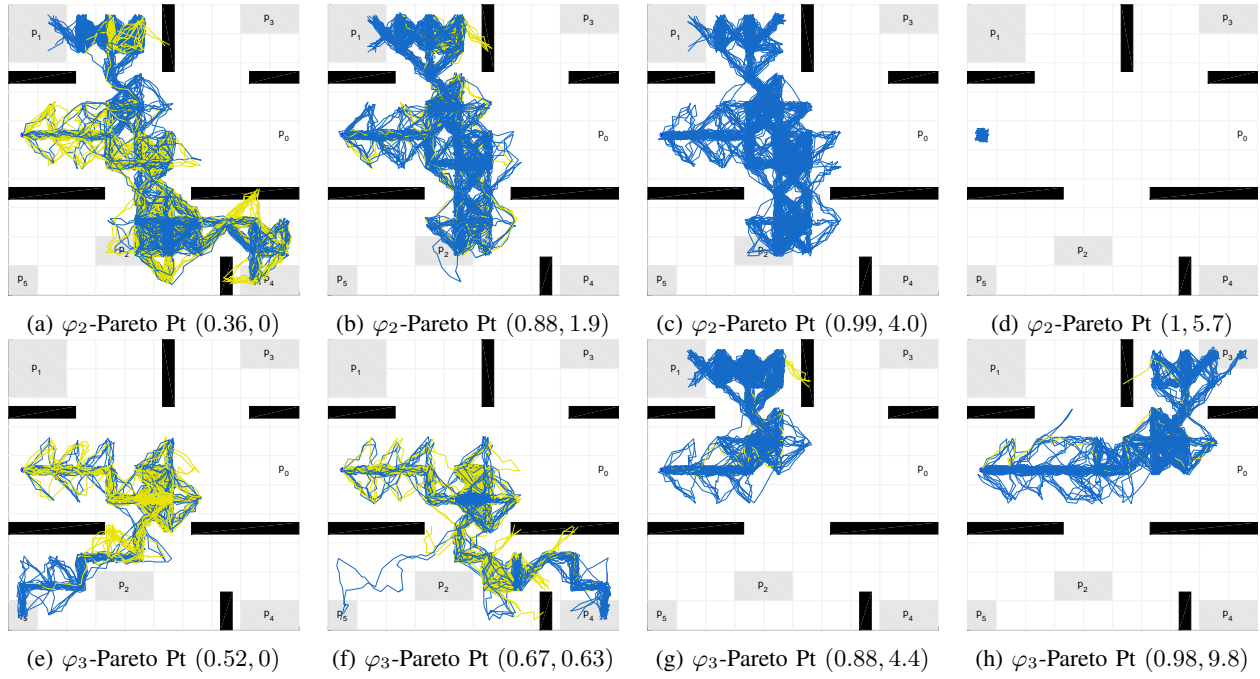


Fig. 9: 50 sample paths under control policies corresponding to points on  $\varphi_2$  and  $\varphi_3$  Pareto curves (Fig. 8). The blue and yellow colors correspond to the successful and failing paths, respectively.

- [2] 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.
- [3] M. Rungger, M. Mazo, Jr., and P. Tabuada, "Specification-guided controller synthesis for linear systems and safe linear-time temporal logic," in *Hyb. Sys.: Comp. and Cont.* ACM, 2013, pp. 333–342.
- [4] M. Lahijanian, S. B. Andersson, and C. Belta, "Formal verification and synthesis for discrete-time stochastic systems," *IEEE Tran. on Aut. Cont.*, vol. 60, no. 8, pp. 2031–2045, 2015.
- [5] S. Esmail Zadeh Soudjani and A. Abate, "Aggregation and control of populations of thermostatically controlled loads by formal abstractions," *IEEE Tran. on Cont. Sys. Tech.*, vol. 23, pp. 975–990, 2015.
- [6] M. Kwiatkowska, A. Mereacre, N. Paoletti, and A. Patanè, "Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques," in *Int. Workshop on Hyb. Sys. and Bio.*, vol. 9271, 2015, pp. 119–140.
- [7] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *Int. Conf. Rob. and Aut.*, 2015, pp. 346–352.
- [8] M. Lahijanian, S. B. Andersson, and C. Belta, "Control of Markov decision processes from PCTL specifications," in *American Control Conf.*, 2011, pp. 311–316.
- [9] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *Conf. on Decision and Cont.*, 2011, pp. 532–538.
- [10] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Tran. on Rob.*, vol. 28, no. 2, pp. 396–409, 2012.
- [11] M. Kwiatkowska and D. Parker, "Automated verification and strategy synthesis for probabilistic systems," in *Int. Symp. on Aut. Tech. for Verif. and Anal.*, vol. 8172, 2013, pp. 5–22.
- [12] J. Wang, X. C. Ding, M. Lahijanian, I. C. Paschalidis, and C. Belta, "Temporal logic motion control using actor-critic methods," *Int. J. of Rob. Res.*, vol. 34, no. 10, pp. 1329–1344, 2015.
- [13] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT Press, 1999.
- [14] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Int. Conf. on Hyb. Sys.: Comp. and Cont.* ACM, 2013, pp. 1–10.
- [15] M. Guo and D. Dimarogonas, "Distributed plan reconfiguration via knowledge transfer in multi-agent systems under local ltl specifications," in *Int. Conf. on Rob. and Aut.*, 2014, pp. 4304–4309.
- [16] K. Kim and G. Fainekos, "Minimal specification revision for weighted transition systems," in *Int. Conf. on Rob. Aut.*, 2013, pp. 4068–4074.
- [17] —, "Revision of specification automata under quantitative preferences," in *Int. Conf. on Rob. Aut.*, 2014, pp. 5339–5344.
- [18] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *Int. Conf. on Hyb. Sys.: Comp. and Cont.* ACM, 2013, pp. 353–362.
- [19] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Tran. on Rob.*, vol. 32, no. 3, pp. 538–599, 2016.
- [20] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe LTL specifications," in *Int. Joint Conf. on Artif. Intell.*, 2015, pp. 1587–1593.
- [21] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *J. Artif. Intell. Res.*, vol. 21, pp. 135–191, 2004.
- [22] J. A. Baier, C. Fritz, M. Bienvenu, and S. McIlraith, "Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners," in *Conf. on Artif. Intell.*, 2008, pp. 1509–1512.
- [23] M. Li, Z. She, A. Turrini, and L. Zhang, "Preference planning for markov decision processes," in *Artif. Intell.*, 2015, pp. 3313–3319.
- [24] M. Lahijanian, S. Almagor, D. Fried, L. E. Kavraki, and M. Y. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction," in *Conf. on Artif. Intell.*, 2015, pp. 3664–3671.
- [25] K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis, "Multi-objective model checking of Markov decision processes," *Logical Meth. in Comp. Sci.*, vol. 4, no. 4, pp. 1–21, 2008.
- [26] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Meth. in Sys. Design*, vol. 19, pp. 291–314, 2001.
- [27] L. de Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.
- [28] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *Int. Symp. on Aut. Tech. for Ver. and Anal.*, 2012, pp. 317–332.
- [29] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Int. Conf. on Comp. Aided Verif.*, vol. 6806. Springer, 2011, pp. 585–591.