Planning with Linear Temporal Logic Specifications: Handling Quantifiable and Unquantifiable Uncertainty

Pian Yu*, Yong Li, David Parker, and Marta Kwiatkowska

***Abstract*— This work studies the planning problem for robotic systems under both quantifiable and unquantifiable uncertainty. The objective is to enable the robotic systems to optimally fulfill high-level tasks specified by Linear Temporal Logic (LTL) formulas. To capture both types of uncertainty in a unified modelling framework, we utilise Markov Decision Processes with Set-valued Transitions (MDPSTs). We introduce a novel solution technique for optimal robust strategy synthesis of MDPSTs with LTL specifications. To improve efficiency, our work leverages limit-deterministic Büchi automata (LDBAs) as the automaton representation for LTL to take advantage of their efficient constructions. To tackle the inherent nondeterminism in MDPSTs, which presents a significant challenge for reducing the LTL planning problem to a reachability problem, we introduce the concept of a Winning Region (WR) for MDPSTs. Additionally, we propose an algorithm for computing the WR over the product of the MDPST and the LDBA. Finally, a robust value iteration algorithm is invoked to solve the reachability problem. We validate the effectiveness of our approach through a case study involving a mobile robot operating in the hexagonal world, demonstrating promising efficiency gains.**

I. Introduction

Uncertainty in planning can be categorised into two types based on the effects of actions: probabilistic and nondeterministic. In probabilistic planning, uncertainty is quantified using probabilities, with Markov Decision Processes (MDPs) and their generalisations serving as the standard modelling frameworks [1], [2]. Nondeterministic planning, on the other hand, addresses unquantifiable uncertainty (such as ambiguity and adversarial environments), typically exploiting the fully observable nondeterministic domain (FOND) as a modelling framework [3]–[5]. Both probabilistic and nondeterministic planning have been extensively studied, leading to significant advances in the field.

Robotic systems are susceptible to many different types of uncertainty, such as sensing and actuation noise, unpredictability in a robot's perception, and dynamic environments [6], [7]. Some sources of uncertainty, such as sensing and actuation noise, can be quantified probabilistically using statistical methods. However, ambiguous uncertainties, such as unpredictable perception and dynamic environments, are often more challenging to quantify. Existing works in robot planning mainly focus on addressing either quantifiable or unquantifiable uncertainty. However, in scenarios such as human-robot collaboration [8], both quantifiable and unquantifiable uncertainties are present. Quantifiable uncertainties may arise from robotic actuation errors, while unquantifiable uncertainties often stem from the unpredictable nature of human behavior. To the best of our knowledge, approaches that can effectively handle both types of uncertainty simultaneously have, however, been less explored. In light of this, we propose to utilise MDPs with set-valued transitions (MDPSTs) [9], [10] as our unified modelling framework. They are attractive because they admit a simplified Bellman equation compared to (more general) MDPs with imprecise probabilities (MDPIPs) [11]–[13] and Uncertain MDPs (UMDPs) [14]–[16], and thus stochastic games [17].

Recently, MDPSTs have been employed to formalise the *trembling-hand problem in nondeterministic domains* [18], where the term "trembling-hand" refers to the phenomenon in which an agent, due to faults or imprecision in its action selection mechanism, may mistakenly perform unintended actions with a certain probability, potentially leading to goal failures. Specifically, this approach demonstrates that the human-robot co-assembly problem can be modelled using MDPSTs, yielding more efficient solution techniques compared to the stochastic game formulation. In this work, Linear Temporal Logic on finite traces (LTL_f) [19] was used as the task specification language. LTL_f shares the same syntax as Linear Temporal Logic (LTL) [20] but is interpreted over *finite* rather than infinite traces [19]. However, in many robotic applications, such as persistent surveillance and repetitive supply delivery, it is necessary to define the robot's tasks over *infinite* trajectories. Indeed, LTL has been widely applied in robotics research for specifying complex temporal objectives over infinite traces, such as [21]–[26], including MDPs with LTL objectives, e.g., [27]–[29]. A typical approach involves converting the LTL specification into a Deterministic Rabin Automaton (DRA) and then taking the product of the MDP and the DRA [30]. This reduces the problem to a planning problem with a reachability goal over the product space.

In this paper, we propose to formalise the planning problem for robotic systems under both quantifiable and unquantifiable uncertainty with temporal objectives as the strategy synthesis problem for MDPSTs with (full) LTL objectives. We highlight that MDPSTs with LTL objectives

This work was supported by the ERC AdG FUN2MODEL (Grant agreement ID: 834115), ISCAS Basic Research (Grant Nos. ISCAS-JCZD-202406, ISCAS-JCZD-202302), CAS Project for Young Scientists in Basic Research (Grant No. YSBR-040), and ISCAS New Cultivation Project ISCAS-PYFX-202201.

Pian Yu is currently with the Department of Computer Science, University College London (UCL), United Kingdom pian.yu@ucl.ac.uk
*Work was done when she was with the Department of Computer Science, University of Oxford.

David Parker and Marta Kwiatkowska are with the Department of Computer Science, University of Oxford, United Kingdom david.parker, marta.kwiatkowska@cs.ox.ac.uk

Yong Li is with Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China liyong@ios.ac.cn

are studied for the first time in this paper. Due to the presence of unquantifiable uncertainty in MDPSTs, computing the end components [31] of an MDPST becomes nontrivial. As a result, the conventional procedure for MDPs with LTL objectives based on conversion to DRAs does not apply, necessitating new solution techniques developed in this work.

The main contributions are summarised as follows. (i) We propose using MDPSTs as the modelling framework for robot planning under both quantifiable and unquantifiable uncertainty. Although the model has received relatively little study since it was initially proposed in 2007 [9], recent findings highlight its advantages, particularly in terms of computational efficiency [18]. (ii) A novel solution technique is proposed for optimal robust strategy synthesis for MDPSTs with LTL specifications. This technique addresses the inherent nondeterminism in MDPSTs, which complicates the reduction of the LTL planning problem to a reachability problem, by introducing the concept of a Winning Region (WR). To further improve efficiency, we leverage limit-deterministic Büchi automata (LDBAs) [32]–[34], which are typically smaller than conventional DRAs thanks to their efficient construction from LTL. We devise an algorithm for computing WR over the product of the MDPST and the LDBA, and its correctness is formally demonstrated.

II. Preliminaries

This section provides preliminaries for LTL [20] and its equivalent LDBA [34] representation.

LTL [20] extends propositional logic with temporal operators. The syntax of an LTL formula over a finite set of propositions $\mathtt{Prop}$ is defined inductively as:

$$\varphi ::= \mathsf{true} | p \in \mathtt{Prop} | \neg\varphi | \varphi \wedge \varphi | \varphi \vee \varphi | \bigcirc \varphi | \varphi \mathsf{U} \varphi, \qquad (1)$$

where $\bigcirc$ (Next) and U (Until) are temporal operators. As usual, additional Boolean and temporal operators are derived as follows: $\varphi_1 \Rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$ (Implies), $\Diamond\varphi \equiv \mathsf{true}\mathsf{U}\varphi$ (Eventually), and $\Box\varphi = \neg(\Diamond\neg\varphi)$ (Always). The detailed semantics of LTL can be found in [20], [30].

A *trace* $\pi = \pi_0\pi_1\ldots$ is a finite or infinite sequence of propositional interpretations (sets), where for every $i \geq 0$, $\pi_i \in 2^{\mathtt{Prop}}$ is the i-th interpretation in π. Intuitively, π_i is interpreted as the set of propositions that are true at instant i. For a finite trace $\pi \in (2^{\mathtt{Prop}})^*$, we denote the interpretation at the last instant (i.e., index) by $\mathsf{lst}(\pi)$, and we write $\pi \models \varphi$ when an infinite trace $\pi \in (2^{\mathtt{Prop}})^\omega$ satisfies LTL formula φ. The language of φ, denoted by $\mathtt{L}(\varphi)$, is the set of infinite traces over $2^{\mathtt{Prop}}$ that satisfy φ.

Every LTL formula φ over $\mathtt{Prop}$ can be translated into a *nondeterministic Büchi automaton* (NBA) $\mathcal{A}$ [35] over the alphabet $\Sigma = 2^{\mathtt{Prop}}$ that recognises the language $\mathtt{L}(\varphi)$.

Definition 1. *An NBA $\mathcal{A}$ is defined as a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathtt{Acc})$, where Q is the set of states, q_0 is the initial state, $\mathtt{Acc} \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \mapsto 2^Q$ is the nondeterministic transition function.*

A *run* ρ of $\mathcal{A}$ over an infinite trace $w_0w_1\cdots \in \Sigma^\omega$ is an infinite sequence $r_0r_1\cdots \in Q^\omega$ of states such that $r_0 = q_0$ and, for all $i \geq 0$, we have $r_{i+1} \in \delta(r_i, w_i)$. We denote by $\mathtt{Inf}(\rho)$ the set of states that appear infinitely often in the run ρ. A run ρ of $\mathcal{A}$ is called *accepting* if $\mathtt{Inf}(r) \cap \mathtt{Acc} \neq \emptyset$. The language of $\mathcal{A}$, denoted $\mathtt{L}(\mathcal{A})$, is the set of all traces that have an accepting run in $\mathcal{A}$.

NBAs, in general, cannot be used for quantitative analysis of probabilistic systems. Recently, a class of NBAs called limit-deterministic Büchi automata (LDBAs), under mild constraints, have been applied for the quantitative analysis of MDPs [32]–[34]. We will also use the LDBAs constructed by [34] for our planning problem.

Definition 2 (LDBA [34])**.** *An LDBA $\mathcal{A}$ is defined as a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathtt{Acc})$ where*

- *$Q = Q_i \cup Q_{acc}$ is the set of states partitioned into two disjoint sets Q_i and Q_{acc},*
- *$q_0 \in Q_i$ is the initial state,*
- *$\mathtt{Acc} \subseteq Q_{acc}$ is the set of accepting states, and*
- *$\delta = \delta_i \cup \delta_j \cup \delta_{\mathtt{Acc}}$ where $\delta_i : Q_i \times \Sigma \mapsto Q_i, \delta_{acc} : Q_{acc} \times \Sigma \mapsto Q_{acc}$ and $\delta_j : Q_i \times \{\epsilon\} \mapsto 2^{Q_{acc}}$.*

By Definition 2, the LDBAs considered here are deterministic within Q_i and Q_{acc} components; the only nondeterminism lies in the ϵ-transition jumps from Q_i-states to Q_{acc}-states via δ_j function. Note that the ϵ-transitions do not consume a letter from Σ: they are just explicit representations of the nondeterministic jumps in the runs of $\mathcal{A}$. To be accepting in $\mathcal{A}$, a run has to eventually make a nondeterministic jump through δ_j since all accepting states reside only in Q_{acc}. It is easy to translate an LTL formula φ to an LDBA $\mathcal{A}$ such that $\mathtt{L}(\mathcal{A}) = \mathtt{L}(\varphi)$ using state-of-the-art tools such as Owl [34] and Rabinizer 4 [36].

III. Markov decision processes with set-valued transitions

This section introduces Markov Decision Processes with Set-valued Transitions (MDPSTs) [9], [10] as the modelling framework for robot planning under quantifiable and unquantifiable uncertainty. Compared to the definition in [9], we further introduce a labelling function that associates the states of the MDPST with the propositions of an LTL formula.

Definition 3 (MDPSTs)**.** *A MDPST $\mathcal{M}$ is a tuple $(S, s_0, A, \mathcal{F}, \mathcal{T}, \mathcal{L})$, where*

- *S is a finite set of states;*
- *$s_0 \in S$ is the initial state;*
- *A is a finite set of actions;*
- *$\mathcal{F} : S \times A \nrightarrow 2^{2^S}$ is the set-valued nondeterministic state transition (partial) function;*
- *$\mathcal{T} : S \times A \times 2^S \mapsto (0, 1]$ is the transition probability (or mass assignment) function, i.e., given a set $\Theta \in \mathcal{F}(s, a)$, $\mathcal{T}(s, a, \Theta) = \Pr(\Theta|\mathrm{s}, \mathrm{a})$,*
- *$\mathcal{L} : S \to 2^{\mathtt{Prop}}$ is the proposition labelling function, where $\mathtt{Prop}$ is a finite set of propositions.*

Traditional MDPs are, in fact, a special type of MDPSTs, where $\mathcal{T}$ only maps a state and an action to a probabilistic distribution over S instead of the powerset 2^S. As usual, we use $A(s) \subseteq A$ to denote the set of actions *applicable*

at state s. Note that, in MDPSTs, the transition function $\mathcal{F}(s,a)$ returns a set of state sets, i.e., $\mathcal{F}(s,a) \subseteq 2^{\mathcal{S}}$, and the transition probability function $\mathcal{T}$ expresses the probability of transitioning to such sets via a given action.

A path ξ of $\mathcal{M}$ is a finite or infinite sequence of alternating states and actions $\xi = s_0 a_0 s_1 a_1 \cdots$, ending with a state if finite, such that for all $i \geq 0$, $a_i \in A(s_i)$ and $s_{i+1} \in \Theta_i$ for some set $\Theta_i \in \mathcal{F}(s_i, a_i)$. We denote by FPaths (FPaths_s) and IPaths (IPaths_s) the set of all finite and infinite paths of $\mathcal{M}$ (starting from state s), respectively. For a path $\xi = s_0 a_0 s_1 a_1 \cdots$ of $\mathcal{M}$, the sequence $\mathcal{L}(\xi) = \mathcal{L}(s_0)\mathcal{L}(s_1), \cdots$ over $\mathtt{Prop}$ is called the *trace* induced by ξ over $\mathcal{M}$.

A strategy σ of $\mathcal{M}$ is a function $\sigma : \mathrm{FPaths} \to \mathsf{Distr}(\mathrm{A})$ such that, for each $\xi \in \mathrm{FPaths}$, $\sigma(\xi) \in \mathsf{Distr}(A(\mathsf{lst}(\xi)))$, where $\mathsf{lst}(\xi)$ is the last state of the finite path ξ and $\mathsf{Distr}(A)$ denotes the set of all possible distributions over A. Let $\Omega_\sigma^{\mathcal{M}}(s)$ denote the subset of (in)finite paths of $\mathcal{M}$ that correspond to strategy σ and initial state s. Let $\Pi_{\mathcal{M}}$ be the set of all strategies.

Let us now motivate MDPSTs for robot planning under uncertainty with a running example.

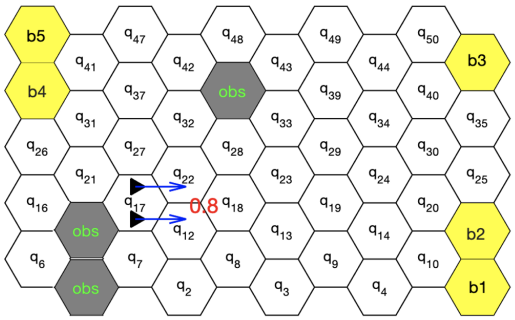

Fig. 1: Hexagonal world.

Example 1 (Hexagonal world)**.** *We consider a hexagonal grid map, as shown in Fig. 1. Hexagonal grid map representations offer several advantages over quadrangular grid maps, including lower quantisation error [37] and enhanced performance in cooperative robot exploration [38]. The yellow regions, labelled as* $\mathrm{b1}, \mathrm{b2}, \cdots, \mathrm{b5}$*, are base stations, and the grey regions, labelled as* obs*, are obstacle regions. The state of the robot is defined as* (q_i, w)*, where* q_i *represents the specific region where the robot is located and* $w \in \{N, S, E, W\}$ *represents the orientation of the robot. There are 4 action primitives* $\{\mathrm{FR}, \mathrm{BK}, \mathrm{TR}, \mathrm{TL}\}$*, which stand for move forward, move backward, turn right, and turn left, respectively. The robot's motion is subject to uncertainty due to actuation noise and drifting. It is known that the probabilities of the 4 actions being executed correctly are* $0.8, 0.7, 0.9,$ *and* 0.9*, respectively. Moreover, it should be noted that, depending on the precise location (which is not available due to imprecise sensing/perception) within each region and the orientation of the robot, there may be several potential target states when the action is correctly executed. For instance, as depicted in Fig. 1, when the robot is at state* (q_{17}, E) *and wants to take an action* FR*, with probability 0.8 it ends up in state* (q_{12}, E) *or* (q_{22}, E)*.*

In this example, it is convenient to abstract the robot dynamics as an MDPST since one can combine the moves of all potential target regions as a set-valued transition for each correctly executed motion. For instance, when the robot's state is (q_{17}, E) *and it takes an action* FR*, then there are three possible transitions: i) with probability 0.8 it moves to the set* $\{(q_{12}, E), (q_{22}, E)\}$*, ii) with probability 0.1 it moves to the singleton set* $\{(q_{27}, E)\}$*, and iii) with probability 0.1 it moves to the singleton set* $\{(q_7, E)\}$*.*

Previously, MDPSTs were used to formulate the trembling-hand problem in nondeterministic domains [18], where LTL_f was utilised as the specification language. Note that, for LTL_f objectives, one can translate the formula into a deterministic finite automaton. For LTL goals, however, this is more involved because DFAs are not sufficient, and one has to resort to automata over infinite traces. This complicates the strategy synthesis procedure and requires us to develop new solution techniques based on the Winning Region (see Def. 10) to capture acceptance conditions.

Let us first recap the definitions of a *feasible distribution* and *nature* for MDPSTs originally introduced in [18].

Given an MDPST $\mathcal{M}$, define the set of reachable states of (s,a) as $\mathsf{Post}_{\mathcal{M}}(s,a) = \{s' \mid \exists \Theta \in \mathcal{F}(s,a) \text{ s.t. } \mathcal{T}(s,a,\Theta) > 0 \wedge s' \in \Theta\}$. A *feasible distribution* of $\mathcal{M}$ guarantees that, given a state-action pair (s,a), *(i)* the sum of probabilities of selecting a state from $\mathsf{Post}_{\mathcal{M}}(s,a)$ equals 1; *(ii)* the sum of probabilities of selecting a state from a set $\Theta \in \mathcal{F}(s,a)$ equals $\mathcal{T}(s,a,\Theta)$; and *(iii)* the probability of selecting a state outside $\mathsf{Post}_{\mathcal{M}}(s,a)$ is 0. In the following definition, $\iota_{s'}^{\Theta}$ indicates whether s' is *in* Θ. Hence $\iota_{s'}^{\Theta} = 1$ if $s' \in \Theta$ and $\iota_{s'}^{\Theta} = 0$ otherwise. Furthermore, $\alpha_{s'}^{\Theta}$ represents the probability of s' being *selected* from Θ if $s' \in \Theta$. Note that Definition 5 in [18] only allows deterministic choice within Θ (i.e., $\alpha_{s'}^{\Theta} = 1$ if s' is selected, and 0 otherwise). In this work, we also permit probabilistic choice within Θ.

Definition 4 (Feasible distribution in MDPSTs)**.** *Let* $\mathcal{M} = (S, s_0, A, \mathcal{F}, \mathcal{T}, \mathcal{L})$ *be an MDPST and* (s,a) *a state-action pair, where* $a \in A(s)$*.* $\mathfrak{h}_s^a \in \mathsf{Distr}(S)$ *is a* feasible distribution *of* (s,a)*, denoted by* $s \xrightarrow{a} \mathfrak{h}_s^a$*, if*

(i) $\sum_{s' \in \Theta} \alpha_{s'}^{\Theta} = 1$, *for* $\Theta \in \mathcal{F}(s,a)$*;*
(ii) $\mathfrak{h}_s^a(s') = \sum_{\Theta \in \mathcal{F}(s,a)} \iota_{s'}^{\Theta} \alpha_{s'}^{\Theta} \mathcal{T}(s,a,\Theta)$, *for* $s' \in \mathsf{Post}_{\mathcal{M}}(s,a)$*;*
(iii) $\mathfrak{h}_s^a(s') = 0$, *for* $s' \in S \setminus \mathsf{Post}_{\mathcal{M}}(s,a)$*.*

A *nature* is defined to characterize the unquantifiable uncertainty in MDPSTs, motivated by the definition of *nature* in robust MDPs [14]. One can think of nature as the strategy controlled by the adversarial environment.

Definition 5 (Nature for MDPSTs [18])**.** *A* nature *of an MDPST is a function* $\gamma : \mathrm{FPaths} \times \mathrm{A} \to \mathsf{Distr}(\mathrm{S})$ *such that* $\gamma(\xi, a) \in \mathcal{H}_s^a$ *for* $\xi \in \mathrm{FPaths}$ *and* $a \in A(\mathsf{lst}(\xi))$*, where* $\mathcal{H}_s^a$ *is the set of feasible distributions of* (s,a)*.*

Suppose we fix a nature γ. The probability of an agent

strategy σ satisfying an LTL specification φ is denoted by

$$\Pr^{\sigma,\gamma}_{\mathcal{M}}(\varphi) := \Pr_{\mathcal{M}}(\{\xi \in \Omega^{\mathcal{M}}_{\sigma,\gamma}(s_0) \mid \mathcal{L}(\xi) \models \varphi\}),$$

where $\Omega^{\mathcal{M}}_{\sigma,\gamma}(s_0)$ is the set of all probable paths generated by the agent strategy σ and nature γ from initial state s_0.

Similarly to Definitions 7-8 in [18], we now define (optimal) robust strategies for MDPSTs with LTL specifications, rather than LTL_f, which account for all possible natures. Of course, given a fixed strategy σ and a nature γ, one can deduce a Markov chain $\mathcal{M}_{\sigma,\gamma}$ from $\mathcal{M}$.

Definition 6 (Robust strategy)**.** *Let $\mathcal{M}$ be an MDPST, φ an LTL formula, and $\beta \in [0,1]$ a threshold. An agent strategy σ* robustly enforces *φ in $\mathcal{M}$ wrt β if, for every nature γ, the probability of generating paths satisfying φ in $\mathcal{M}$ is no less than β, that is, $P^{\sigma}_{\mathcal{M}}(\varphi) \geq \beta$, where $P^{\sigma}_{\mathcal{M}}(\varphi) := \min_{\gamma}\{\Pr^{\sigma,\gamma}_{\mathcal{M}}(\varphi)\}$. Such a strategy σ is referred to as a robust strategy for $\mathcal{M}$ (with respect to β).*

Definition 7 (Optimal robust strategy)**.** *An optimal strategy σ^* that robustly enforces an LTL formula φ in an MDPST $\mathcal{M}$ is given by $\sigma^* = \arg\max_{\sigma}\{\Pr^{\sigma}_{\mathcal{M}}(\varphi)\}$. In this case, σ^* is referred to as an* optimal robust strategy *for $\mathcal{M}$.*

The optimal robust strategy synthesis problem considered in this paper is formulated as follows.

Problem 1 (Optimal Robust Strategy Synthesis)**.** *Given an MDPST $\mathcal{M}$ and an LTL formula φ, synthesise an optimal robust strategy σ^*.*

IV. Solution technique

We now introduce our solution to Problem 1 for a given MDPST $\mathcal{M}$ and LTL formula φ. Our approach is based on a reduction to the reachability problem, but is more challenging than for MDPs because of unquantifiable uncertainty, and than for LTL_f because of the need to consider infinite paths. It contains 3 steps.

1) Construct the product $\mathcal{M}^{\times}$ of the MDPST $\mathcal{M}$ and the LDBA $\mathcal{A}$ derived from the LTL specification φ.
2) Reduce Problem 1 to a reachability problem over the product MDPST $\mathcal{M}^{\times}$. This step presents a significant challenge; we address it by introducing the concept of a Winning Region for MDPSTs, along with a novel algorithm for computing it.
3) Synthesise the strategy of the reachability problem over the product $\mathcal{M}^{\times}$.

We will introduce each step in detail below.

A. Product MDPST

As mentioned before, we first obtain an LDBA $\mathcal{A} = (Q, q_0, \Sigma, \delta = \delta_i \cup \delta_j \cup \delta_{acc}, \texttt{Acc})$ from the LTL formula φ using state-of-the-art tools such as Rabinizer 4 [36]. We prefer LDBAs over DRAs because nondeterministic LDBAs are usually smaller than DRAs [34]. This thus yields smaller product MDPSTs and smaller strategies. Then, we construct the product MDPST $\mathcal{M}^{\times}$ of the MDPST $\mathcal{M} = (S, s_0, A, \mathcal{F}, \mathcal{T}, \mathcal{L})$ and the LDBA $\mathcal{A}$ as follows.

Definition 8 (Product MDPST)**.** *A product MDPST is a tuple $\mathcal{M}^{\times} = (S^{\times}, s^{\times}_0, A^{\times}, \mathcal{F}^{\times}, \mathcal{T}^{\times}, \mathcal{L}^{\times}, \texttt{Acc}^{\times})$, where $S^{\times} = S \times Q$ is the set of states, $s^{\times}_0 = (s_0, q_0)$ is the initial state, and*

- *$A^{\times} = A \cup A^{\epsilon}$ where $A^{\epsilon} := \{\epsilon_q \mid q \in Q\}$;*
- *$\mathcal{F}^{\times} : S^{\times} \times A^{\times} \nrightarrow 2^{2^{S^{\times}}}$ is the set-valued nondeterministic transition function. For every $a \in A^{\times}$, $(s,q) \in S^{\times}$, we define $\Theta^{\times} \in \mathcal{F}^{\times}((s,q),a)$ as follows:*
  - i) *if $a \in A$, let $q' = \delta(q, \mathcal{L}(s))$, and define $\Theta^{\times} = \{(s', q') \mid s' \in \Theta\}$, for every $\Theta \in \mathcal{F}(s,a)$;*
  - ii) *otherwise $a = \epsilon_{q'} \in A^{\epsilon}$, then for every $q' \in \delta_j(q, \epsilon)$, define $\Theta^{\times} = \{(s, q')\}$.*
- *$\mathcal{T}^{\times} : S^{\times} \times A^{\times} \times 2^{S^{\times}} \mapsto [0,1]$, where*
  - *$\mathcal{T}^{\times}((s,q),a,\Theta^{\times}) = \mathcal{T}(s,a,\Theta)$, if $a \in A(s), \Theta^{\times} = \Theta \times \{q'\}$ for $\Theta \in \mathcal{F}(s,a)$ where $q' = \delta(q, \mathcal{L}(s))$;*
  - *$\mathcal{T}^{\times}((s,q),a,\Theta^{\times}) = 1$ if $a \in A^{\epsilon}, \Theta^{\times} = \{(s,q')\}$ for some $q' \in \delta_j(q,\epsilon)$ with $a = \epsilon_{q'}$,*
  - *$\mathcal{T}^{\times}((s,q),a,\Theta^{\times}) = 0$, otherwise.*
- *$\mathcal{L}^{\times} : S^{\times} \to 2^{Prop}$, where $\mathcal{L}^{\times}((s,q)) = \mathcal{L}(s)$;*
- *$\texttt{Acc}^{\times} = \{(s,q) \in S^{\times} \mid q \in \texttt{Acc}\}$.*

An infinite path $\xi^{\times}$ of $\mathcal{M}^{\times}$ satisfies the Büchi condition if $\texttt{Inf}(\xi^{\times}) \cap \text{Acc}^{\times} \neq \emptyset$. Such a path is said to be accepting.

When an MDPST action $a \in A$ is taken in the product MDPST $\mathcal{M}^{\times}$, the alphabet used to transition the LDBA is deduced by applying the proposition labelling function to the current MDP state: $\mathcal{L}(s) \in 2^{\texttt{Prop}}$. In this case, the LDBA transition $\delta(q, \mathcal{L}(s))$ is deterministic. Otherwise, if an ϵ-transition $\epsilon_{\hat{q}} \in \{\epsilon_q \mid q \in Q\}$ is taken, the LDBA selects an ϵ-transition, and the nondeterminism of $\delta_j(q,\epsilon)$ is resolved by transitioning the automaton state to $\hat{q}$.

B. Reduction to a reachability problem

Conventional planning approaches for MDPs against LTL specification require computing *maximal end components* (MECs) in the product and determining which MECs are accepting. These MECs are then regarded as the goal states in the reachability planning problems. For MDPSTs, however, this approach is not applicable. To better understand this, let's first review the definition of (maximal) end-components ((M)EC) for MDPs.

Definition 9 (EC for MDPs [30], [31])**.** *An* end component (EC) *of an MDP $\mathcal{M}$ is a sub-MDP $\mathcal{M}'$ of $\mathcal{M}$ such that its underlying graph is strongly connected. A* maximal EC (MEC) *is maximal under set inclusion.*

Lemma 1 (EC properties for MDPs. Theorems 3.1 and 4.2 of [31])**.** *Once an end component E of an MDP $\mathcal{M}$ is entered, there is a strategy that i) visits every state-action pair in E infinitely often with probability 1, and ii) stays in E forever.*

Lemma 1 makes it possible to reduce a planning problem over MDPs with LTL objectives to a reachability problem over the product MDP. This is due to the fact that, whenever a state $s^{\times}$ of an accepting MEC E of the product MDP $\mathcal{M}^{\times}$ is reached, there exists a strategy of $\mathcal{M}^{\times}$ starting from $s^{\times}$ that ensures every state in E (including the accepting

Algorithm 1 *Compute winning region for MDPST*

Require: Product MDPST $\mathcal{M}^{\times}$.
Ensure: Winning region $W^{\times}$.
1: Compute S_p and construct sub-MDPST $\mathcal{M}^{\times}_{sub} = (S_p, s_0^{\times}, A^{\times}, \mathcal{F}_p, \mathcal{T}_p, \mathcal{L}_p, \texttt{Acc}^{\times})$;
2: flag $= 1$;
3: **while** flag $= 1$ and $\texttt{Acc}^{\times} \neq \emptyset$ **do**
4: Split $\texttt{Acc}^{\times}$ into two virtual copies $I_{in} = \{s^{in} : s^{in}$ is a virtual copy of $s, \forall s \in \texttt{Acc}^{\times}\}$ and $I_{out}\{s^{in} : s^{out}$ is a virtual copy of $s, \forall s \in \texttt{Acc}^{\times}\}$;
5: $\hat{S} = (S_p \setminus \texttt{Acc}^{\times}) \cup I_{\text{in}} \cup I_{\text{out}}$;
6: Construct the MDPST $\hat{\mathcal{M}}^{\times}_{sub}$ (cf. (5)) over $\hat{S}$;
7: Compute the optimal value function V_{sat} for $\hat{\mathcal{M}}^{\times}_{sub}$ with the robust dynamic programming operator T in (3);
8: **if** $\exists s^{out} \in I_{out}$ s.t. $V_{sat}(s^{out}) \neq 1$ **then**
9: flag $\leftarrow 1$;
10: Update S_p and $\texttt{Acc}^{\times}$;
11: **else**
12: flag $\leftarrow 0$;
13: **end if**
14: **end while**
15: $W^{\times} = S_p$.

states) will be visited infinitely often (according to Lemma 1), thereby satisfying the LTL objective.

For EC decomposition, MDPs can be seen as directed graphs where each state corresponds to a node and each action-labelled (probabilistic) transition corresponds to an edge. However, this approach cannot be directly applied to MDPSTs where transitions lead to set-valued successors rather than individual states. For instance, consider a set-valued transition $\Theta^{\times} = \{s', s'', s'''\} \in \mathcal{F}^{\times}(s^{\times}, a)$ in a product MDPST $\mathcal{M}^{\times}$. Here it is not sufficient to add edges for all pairs $(s^{\times}, s'), (s^{\times}, s''), (s^{\times}, s''')$, as the adversarial nature may prevent reaching some states in $\Theta^{\times}$ from $s^{\times}$. This poses a significant challenge in identifying the set of states in the product MDPST $\mathcal{M}^{\times}$ that are guaranteed to visit the set of accepting states $\texttt{Acc}^{\times}$ infinitely often with probability 1, an essential step in reducing the LTL planning problem to a reachability problem. To address this challenge, we propose a procedure for identifying a set of states, called the Winning Region, in an MDPST that are guaranteed to visit a set of accepting states infinitely often with probability 1, defined formally below.

Definition 10 (Winning Region for MDPSTs)**.** *Given an MDPST $\mathcal{M} = (S, s_0, A, \mathcal{F}, \mathcal{T}, \mathcal{L})$ with a set of accepting states $\texttt{Acc} \subseteq S$, we say a set of states $W \subseteq S$ is a Winning Region (WR) for the MDPST $\mathcal{M}$ if, for every state $s \in W$, there exists a strategy $\sigma(s)$ starting from s such that $\Pr^{\sigma(s)}_{\mathcal{M}}(\square\lozenge\texttt{Acc}) = 1$.*

Next, we propose an algorithm for computing the WR $W^{\times}$ of the product MDPST $\mathcal{M}^{\times}$, which is outlined in Algorithm 1. The algorithm consists of the following steps.

First, we introduce an optimisation that computes a sub-MDPST $\mathcal{M}^{\times}_{sub}$ of $\mathcal{M}^{\times}$, which includes only states that are (forward) reachable from the initial state $s_0^{\times}$ and (backward) reachable from the set of accepting states $\texttt{Acc}^{\times}$ (line 1). Denote by i) $S_p \subseteq S^{\times}$ the set of states that can be reached from both the initial and accepting states and ii) $\mathcal{M}^{\times}_{sub} = (S_p, s_0^{\times}, A^{\times}, \mathcal{F}_p, \mathcal{T}_p, \mathcal{L}^{\times})$ the sub-MDPST constructed from $\mathcal{M}^{\times}$ with respect to S_p (an algorithm for computing S_p and $\mathcal{M}^{\times}_{sub}$ can be found in [18]).

Second, we iteratively remove states in S_p that cannot visit $\texttt{Acc}^{\times}$ infinitely often with probability 1 (lines 2-20). Before starting the iteration, a flag is set to 1 (line 2), indicating that the iteration should proceed. Each iteration begins by splitting the set of accepting states $\texttt{Acc}^{\times}$ into two virtual copies: i) I_{in}, which only has incoming transitions into $\texttt{Acc}^{\times}$, and ii) I_{out}, which only has outgoing transitions from $\texttt{Acc}^{\times}$ (line 4). Then a new state space can be defined as $\hat{S} := (S_p \setminus \texttt{Acc}^{\times}) \cup I_{in} \cup I_{out}$ (line 5).

Over $\hat{S}$, we can construct a new product MDPST

$$\hat{\mathcal{M}}^{\times}_{sub} = (\hat{S}, s_0^{\times}, \hat{A}^{\times}, \hat{\mathcal{F}}^{\times}, \hat{\mathcal{T}}^{\times}, \hat{\mathcal{L}}^{\times}) \tag{2}$$

which is equivalent to $\mathcal{M}^{\times}_{sub}$ (line 6). For each copy $s^{\text{in}} \in I_{in}$ of an accepting state $s \in \texttt{Acc}^{\times}$, we assign only a self-loop transition. As a result, each time s^{in} is visited, it will be visited infinitely often. Due to space limitations, we refer to [39] for the detailed construction of $\hat{\mathcal{M}}^{\times}_{sub}$.

We now give a robust value iteration algorithm [14] for computing the robust maximal probability of reaching I_{in} from each state $s^{\times} \in \hat{S}$ (line 7). Define a value function $V_{sat} : \hat{S} \to \mathbb{R}_{\geq 0}$, where

$$V_{sat}(s^{\times}) = \max_{\sigma^{\times}(s^{\times})} \min_{\gamma^{\times}} \Big\{ \Pr_{\hat{\mathcal{M}}^{\times}_{sub}}(\{\xi^{\times} \in \Omega^{\hat{\mathcal{M}}^{\times}_{sub}}_{\sigma^{\times}(s^{\times}),\gamma^{\times}}(s^{\times}) \mid \hat{\mathcal{L}}^{\times}(\xi^{\times}) \models \lozenge I_{in}\}) \Big\},$$

which represents the robust maximal probability of reaching I_{in} from $s^{\times}$. Then one can get that $V_{sat}(s^{\times}) = 1, \forall s^{\times} \in I_{in}$.

It was shown in [9], [18] that a simplified Bellman equation exists for MDPSTs. Therefore, for $s^{\times} \in \hat{S} \setminus I_{in}$, the robust dynamic programming operator T can be designed as

$$T(V_{sat})(s^{\times}) = \max_{a \in \hat{A}^{\times}(s^{\times})} \Big\{ \sum_{\Theta \in \hat{\mathcal{F}}^{\times}(s^{\times}, a)} \hat{\mathcal{T}}^{\times}(s^{\times}, a, \Theta) \min_{s' \in \Theta} \{V_{sat}(s')\} \Big\}. \tag{3}$$

Once the robust value iteration converges and thus the optimal value function V_{sat} is obtained, we first check whether there exists a state $s^{out} \in I_{out}$ such that $V_{sat}(s^{out}) \neq 1$. If such a state exists, we set flag to 1, and then remove the corresponding state s from both S_p and $\texttt{Acc}^{\times}$ (lines 8-10). Otherwise, the flag is set to 0 (11-12). The iteration continues if the flag is 1 and terminates once flag becomes 0 or $\texttt{Acc}^{\times} = \emptyset$. Once the iteration terminates, the algorithm returns the WR $W^{\times}$ (line 15). Our main result then follows.

Theorem 1. *Given an MDPST $\mathcal{M}$ and an LTL formula φ, the maximal probability of satisfying φ is given by*

$$\max_{\sigma \in \Pi_{\mathcal{M}}} \{\Pr^{\sigma}_{\mathcal{M}}(\varphi)\} = \max_{\sigma^{\times} \in \Pi_{\mathcal{M}^{\times}}} \{\Pr^{\sigma^{\times}}_{\mathcal{M}^{\times}}(\lozenge \mathrm{W}^{\times})\}, \tag{4}$$

where $W^\times$ is the WR computed by Algorithm 1.

Proof Sketch. We prove Theorem 1 in two steps. First, we show the correctness of Algorithm 1, i.e., that the output $W^\times$ of Algorithm 1 is indeed the WR of the product MDPST $\mathcal{M}^\times$. Then, we prove that (4) holds by verifying both sides of the inequality and subsequently constructing the induced policy on $\mathcal{M}$. □

An example showing the steps of Algorithm 1 and a full proof of Theorem 1 are provided in an extended version [39].

C. Optimal robust strategy synthesis

We have thus reduced Problem 1 to the reachability problem over $\mathcal{M}^\times_{sub}$, where the goal set is given by the WR $W^\times$. For states $s^\times \in W^\times$, it holds that $V_{sat}(s^\times) = 1$. For states $s^\times \in S_p \setminus W^\times$, the optimal value function V_{sat} can be determined by conducting another run of the robust value iteration algorithm (3). The optimal robust strategy $\sigma^\times$ can be derived from V_{sat} using standard methods.

V. Experiments

In this section, a case study is provided to demonstrate the effectiveness of our method. We implemented the solution technique proposed in Section IV in Python, and use Rabinizer 4 [36] for the LTL-to-LDBA construction. For the robust value iteration, we set the convergence threshold to 10^{-3}, i.e., the value iteration stops when $\max_{s\in S_l}\{|V^{k+1}_{sat}(s) - V^k_{sat}(s)|\} < 10^{-3}$. All simulations are carried out on a Macbook Pro (2.6 GHz 6-Core Intel Core i7 and 16 GB of RAM) and the implementation code can be found at: https://github.com/piany/MDPST-full-LTL.

We consider a mobile robot moving in the hexagonal world described in Example 1, where the size of the workspace is denoted by (N_x, N_y). As explained, the robot dynamics can be abstracted as an MDPST. The robot is required to persistently survey three goal regions while avoiding obstacles at all times. This task is expressed as the LTL formula

$$\varphi_{persistavoid} = (\square\lozenge \mathtt{b1} \vee \mathtt{b2}) \wedge (\square\lozenge \mathtt{b3}) \wedge (\square\lozenge \mathtt{b4} \vee \mathtt{b5}) \wedge (\square\neg \mathtt{obs}).$$

The corresponding LDBA derived using Rabinizer 4 has 4 states. For the scenario $(N_x, N_y) = (10, 5)$, the constructed product MDPST $\mathcal{M}^\times$ has 800 states and 5440 (single and set-valued) edges. The WR $W^\times$ is computed using Algorithm 1, which has 509 states. The initial state of the robot is (q_1, N) and one can compute that $\max\{\Pr_{\mathcal{M}^\times}(\varphi_{persistavoid})\} = 0.85$. The (MDPST, LDBA, and product MDPST) model construction took in total 0.467s and the strategy synthesis took 9.144s.

To highlight the computational advantage of LDBA over DRA, we also consider DRA as representations for the LTL task $\varphi_{persistavoid}$. The resulting DRA has 8 states, whereas the LDBA has only 4 states. We compare the performance of both representations in three scenarios: $(N_x, N_y) = (10, 5), (N_x, N_y) = (16, 8)$, and $(N_x, N_y) = (20, 10)$. TABLE I shows the number of states and transitions $(|S^\times|, |\mathcal{T}^\times|)$ of the product MDPST $\mathcal{M}^\times$, along with the model construction time T_{mdl}, and the strategy synthesis time T_{sys} for each scenario (N_x, N_y). The results clearly show that strategy synthesis with LDBA is significantly faster than with DRA, particularly as the workspace size increases.

TABLE I: The number of states and transitions $(|S^\times|, |\mathcal{T}^\times|)$ of the product MDPST $\mathcal{M}^\times$, the model construction time T_{mdl}, and the strategy synthesis time T_{sys} for different automation choices $\mathcal{A}$ and different scenarios (N_x, N_y).

(N_x, N_y)	$\mathcal{A}$	$(\lvert S^\times\rvert, \lvert\mathcal{T}^\times\rvert)$	$T_{mdl}(s)$	$T_{sys}(s)$
(10, 5)	LDBA	(800, 5440)	0.888	10.745
	DRA	(1600, 10880)	0.986	17.363
(16, 8)	LDBA	(2048, 14344)	0.975	88.529
	DRA	(4098, 28688)	1.696	231.343
(20, 10)	LDBA	(3200, 22728)	2.058	288.662
	DRA	(6400, 45456)	1.962	748.126

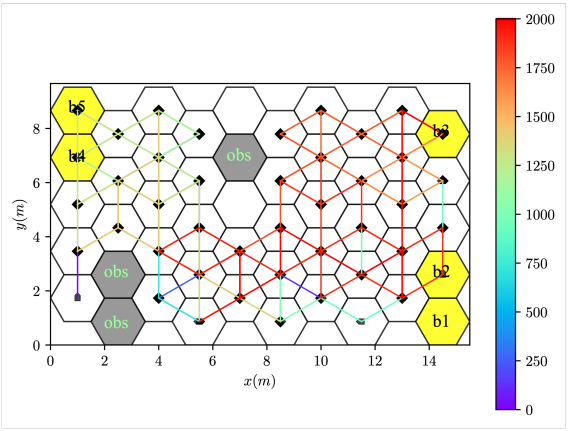

Fig. 2: Simulated trajectory of 2000 time steps for the LTL task $\varphi_{persistavoid}$, where the color bar denotes the time steps.

To verify robustness, we perform 1000 Monte Carlo simulations of 2000 time steps for scenario $(N_x, N_y) = (10, 5)$. For each simulation, we randomly choose the set of parameters $\{\alpha^\theta_{s'} : \theta \in \mathcal{F}^\times(s, a), s' \in \theta\}$ for each state-action pair (s, a), to resolve the uncertainty for the set-valued transitions. We adopt the optimal robust strategy for the strategy prefix and the Round-Robin strategy once the system enters the WR. The task $\varphi_{persistavoid}$ is satisfied 868 times out of the 1000 simulations, which verifies the probabilistic satisfaction guarantee. Fig. 2 depicts one of the simulated trajectories. One can see that the LTL specification $\varphi_{persistavoid}$ is satisfied.

VI. Conclusion

This work studied the robot planning problem under both quantifiable and unquantifiable uncertainty, and subject to high-level LTL task specifications. MDPSTs were proposed as a unified modelling framework for handling both types of uncertainties. Additionally, a sound solution technique was introduced for synthesising the optimal robust strategy for MDPSTs with LTL specifications. For future work, we plan to explore the plan synthesis problem for MDPSTs, subject to both temporal logic specifications and cost constraints.

REFERENCES

[1] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[2] M. Natarajan and A. Kolobov, *Planning with Markov decision processes: An AI perspective*. Springer Nature, 2022.

[3] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "Weak, strong, and strong cyclic planning via symbolic model checking," *Artif. Intell.*, vol. 147, no. 1-2, pp. 35–84, 2003.

[4] M. Ghallab, D. S. Nau, and P. Traverso, *Automated planning - theory and practice*, 2004.

[5] H. Geffner and B. Bonet, *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[6] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics," 2005.

[7] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2011.

[8] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous robots*, vol. 42, pp. 957–975, 2018.

[9] F. W. Trevizan, F. G. Cozman, and L. N. de Barros, "Planning under Risk and Knightian Uncertainty," in *IJCAI*, vol. 2007, 2007, pp. 2023–2028.

[10] F. W. Trevizan, F. G. Cozman, and L. N. De Barros, "Mixed probabilistic and nondeterministic factored planning through Markov decision processes with set-valued transitions," in *Workshop on A Reality Check for Planning and Scheduling Under Uncertainty at ICAPS*, 2008, p. 62.

[11] C. C. White III and H. K. Eldeib, "Markov decision processes with imprecise transition probabilities," *Operations Research*, vol. 42, no. 4, pp. 739–749, 1994.

[12] J. K. Satia and R. E. Lave Jr, "Markovian decision processes with uncertain transition probabilities," *Operations Research*, vol. 21, no. 3, pp. 728–740, 1973.

[13] R. Givan, S. Leach, and T. Dean, "Bounded-parameter Markov decision processes," *Artificial Intelligence*, vol. 122, no. 1-2, pp. 71–109, 2000.

[14] A. Nilim and L. El Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[15] O. Buffet and D. Aberdeen, "Robust planning with (L) RTDP," in *Proceedings of the 19th international joint conference on Artificial intelligence*, 2005, pp. 1214–1219.

[16] E. M. Hahn, V. Hashemi, H. Hermanns, M. Lahijanian, and A. Turrini, "Interval Markov decision processes with multiple objectives: from robust strategies to Pareto curves," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 29, no. 4, pp. 1–31, 2019.

[17] A. Condon, "On algorithms for simple stochastic games," *Advances in computational complexity theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 13, pp. 51–73, 1993.

[18] P. Yu, S. Zhu, G. De Giacomo, M. Kwiatkowska, and M. Vardi, "The trembling-hand problem for LTL_f planning," in *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, 2024, pp. 3631–3641.

[19] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, vol. 13, 2013, pp. 854–860.

[20] A. Pnueli, "The temporal logic of programs," in *FOCS*, 1977, pp. 46–57.

[21] X. Ding, M. Lazar, and C. Belta, "LTL receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[22] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Hierarchical LTL-task mdps for multi-agent coordination through auctioning and learning," *The international journal of robotics research*, 2019.

[23] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4051–4066, 2018.

[24] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of markov decision processes for linear temporal logic specifications," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1091–1096.

[25] M. Wen and U. Topcu, "Probably approximately correct learning in stochastic games with temporal logic specifications." in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 3630–3636.

[26] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *2019 IEEE 58th conference on decision and control (CDC)*. IEEE, 2019, pp. 5338–5343.

[27] M. Kwiatkowska and D. Parker, "Automated verification and strategy synthesis for probabilistic systems," in *Automated Technology for Verification and Analysis: 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*. Springer, 2013, pp. 5–22.

[28] M. Kwiatkowska, G. Norman, and D. Parker, "Probabilistic model checking and autonomy," *Annual review of control, robotics, and autonomous systems*, vol. 5, no. 1, pp. 385–410, 2022.

[29] R. I. Brafman, G. De Giacomo, and F. Patrizi, "LTL_f/LDL_f non-markovian rewards," in *AAAI*, S. A. McIlraith and K. Q. Weinberger, Eds., 2018, pp. 1771–1778.

[30] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[31] L. De Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1998.

[32] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *Journal of the ACM (JACM)*, vol. 42, no. 4, pp. 857–907, 1995.

[33] E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang, "Lazy probabilistic model checking without determinisation," in *26th International Conference on Concurrency Theory, CONCUR 2015*, 2015, pp. 354–367.

[34] S. Sickert, J. Esparza, S. Jaax, and J. Křetínskỳ, "Limit-deterministic Büchi automata for linear temporal logic," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 312–332.

[35] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Inf. Comput.*, vol. 115, no. 1, pp. 1–37, 1994.

[36] J. Křetínskỳ, T. Meggendorfer, S. Sickert, and C. Ziegler, "Rabinizer 4: from LTL to your favourite deterministic automaton," in *International Conference on Computer Aided Verification*. Springer, 2018, pp. 567–577.

[37] K. Jeevan and S. Krishnakumar, "An image steganography method using pseudo hexagonal image," *Int. J. Pure Appl. Math*, vol. 118, no. 18, pp. 2729–2735, 2018.

[38] H. J. Quijano and L. Garrido, "Improving cooperative robot exploration using an hexagonal world representation," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA 2007)*. IEEE, 2007, pp. 450–455.

[39] P. Yu, Y. Li, D. Parker, and M. Kwiatkowska, "Planning with Linear Temporal Logic Specifications: Handling Quantifiable and Unquantifiable Uncertainty," *arXiv preprint arXiv:2502.19603.*, 2025.

APPENDIX

A. Detailed construction of $\hat{\mathcal{M}}^{\times}_{sub}$

Recall that we split the set of accepting states $\texttt{Acc}^{\times}$ into two virtual copies: i) I_{in}, which only has incoming transitions into $\texttt{Acc}^{\times}$, and ii) I_{out}, which only has outgoing transitions from $\texttt{Acc}^{\times}$. Then the new state-space can be defined as

$$\hat{S} := (S_p \setminus \texttt{Acc}^{\times}) \cup I_{in} \cup I_{out}$$

Over $\hat{S}$, one can further construct a new product MDPST

$$\hat{\mathcal{M}}^{\times}_{sub} = (\hat{S}, s^{\times}_0, \hat{A}^{\times}, \hat{\mathcal{F}}^{\times}, \hat{\mathcal{T}}^{\times}, \hat{\mathcal{L}}^{\times}), \tag{5}$$

where

- $\hat{A}^{\times} = A^{\times} \cup \{\tau_0\}$ and τ_0 is a self-loop action. The set of available actions $\hat{A}^{\times}(s)$ is defined as $\hat{A}^{\times}(s) = A^{\times}(s), \forall s \in S_p \setminus \texttt{Acc}^{\times}$, $\hat{A}^{\times}(s^{\text{out}}) = A^{\times}(s), \forall s \in \texttt{Acc}^{\times}$, and $\hat{A}^{\times}(s^{\text{in}}) = \tau_0, \forall s \in \texttt{Acc}^{\times}$, where s^{out} and s^{in} are respectively the virtual copies of $s \in \texttt{Acc}^{\times}$ in I_{in} and I_{out}.
- $\hat{\mathcal{F}}^{\times} : \hat{S} \times \hat{A}^{\times} \nrightarrow 2^{2^{\hat{S}^{\times}}}$ is the set-valued nondeterministic transition function. To define $\hat{\mathcal{F}}^{\times}$, we let $\Phi = \cup_{s \in S_p} \cup_{a \in A^{\times}} \cup_{\Theta \in \mathcal{F}_p(s,a)} \{\Theta\}$ be the set of all possible target (single- or set-valued) states originating from S_p. For each set $\Theta \in \Phi$, we define a *copy* $\hat{\Theta}$ of Θ as i) $\hat{\Theta} = \Theta$ if $\Theta \subseteq S_p \setminus \texttt{Acc}^{\times}$, ii) $\hat{\Theta} = \{s^{\text{in}} : s \in \Theta\}$ if $\Theta \subseteq \texttt{Acc}^{\times}$, and iii) $\hat{\Theta} = \{s : s \in \Theta \wedge s \in S_p \setminus \texttt{Acc}^{\times}\} \cup \{s^{\text{in}} : s \in \Theta \wedge s \in \texttt{Acc}^{\times}\}$, otherwise. Then, one has that
  - if $s \in S_p \setminus \texttt{Acc}^{\times}$, $\hat{\Theta} \in \hat{\mathcal{F}}^{\times}(s,a)$ iff $\Theta \in \mathcal{F}_p(s,a)$;
  - if $s^{\text{out}} \in I_{out}$, $\hat{\Theta} \in \hat{\mathcal{F}}^{\times}(s^{\text{out}},a)$ iff $\Theta \in \mathcal{F}_p(s,a)$;
  - if $s^{\text{in}} \in I_{in}$, $\hat{\mathcal{F}}^{\times}(s^{\text{in}},\tau_0) = s^{\text{in}}$.
- $\hat{\mathcal{T}}^{\times} : \hat{S} \times \hat{A}^{\times} \times 2^{\hat{S}^{\times}} \mapsto (0,1]$ is the transition probability function, given by
  - $\hat{\mathcal{T}}^{\times}(s,a,\hat{\Theta}) = \mathcal{T}_p(s,a,\Theta)$ for $s \in S_p \setminus \texttt{Acc}^{\times}$,
  - $\hat{\mathcal{T}}^{\times}(s^{\text{out}},a,\hat{\Theta}) = \mathcal{T}_p(s,a,\Theta)$ for $s^{\text{out}} \in I_{out}$,
  - $\hat{\mathcal{T}}^{\times}(s^{\text{in}},\tau_0,s^{\text{in}}) = 1$ for $s^{\text{in}} \in I_{in}$.
- $\hat{\mathcal{L}}^{\times} : \hat{S} \to 2^{Prop}$, where $\hat{\mathcal{L}}^{\times}(s) = \mathcal{L}^{\times}(s)$ if $s \in S_p \setminus \texttt{Acc}^{\times}$ and $\hat{\mathcal{L}}^{\times}(s^{\text{out}}) = \hat{\mathcal{L}}^{\times}(s^{\text{in}}) = \mathcal{L}^{\times}(s)$ otherwise.

Note that $\hat{\mathcal{M}}^{\times}_{sub}$ is equivalent to $\mathcal{M}^{\times}_{sub}$ in the sense that there exists a one-to-one correspondence between the transitions of $\hat{\mathcal{M}}^{\times}_{sub}$ and $\mathcal{M}^{\times}_{sub}$.

B. Example for demonstrating Algorithm 1

Example 2. *Consider a product MDPST $\mathcal{M}^{\times}$ with the set of states $S^{\times} = \{S_1, \cdots, S_{10}\}$ and the set of actions $A^{\times} = \{a, b\}$, as shown in Fig. 3. The initial state is $s^{\times}_0 = S_1$ and the set of accepting states is given by $\texttt{Acc}^{\times} = \{S_4, S_5\}$ (marked in green). For the state S_2, both actions a and b are applicable and the associated (action labelled) probabilistic transitions are depicted. For the remaining states, only action a is applicable and the associated probabilistic transitions are depicted (where the action label is omitted). Notice that state S_{10} is not backward reachable from the set of accepting states $\texttt{Acc}^{\times}$, therefore it follows that $S_p = \{S_1, \cdots, S_9\}$ and the resulting sub-MDPST $\mathcal{M}^{\times}_{sub}$ is shown within the shaded box of Fig. 3.*

In the first iteration of Algorithm 1, one starts by splitting $\texttt{Acc}^{\times}$ into two virtual copies of $I_{in} = \{S^{in}_4, S^{in}_5\}$ and $I_{out} = \{S^{out}_4, S^{out}_5\}$. Then the new state space $\hat{S} = \{S_1, \cdots, S_3, S^{in}_4, S^{in}_5, S^{out}_4, S^{out}_5, S_6, \cdots, S_9\}$ and the constructed MDPST $\hat{\mathcal{M}}^{\times}_{sub}$ is drawn in Fig. 4. Let $V_{sat}(S^{in}_4) = V_{sat}(S^{in}_5) = 1$. With the robust dynamic programming operator T in (3), one can compute that $V_{sat}(S_1) = 0.8, V_{sat}(S_2) = V_{sat}(S_3) = V_{sat}(S^{out}_4) = 1$, $V_{sat}(S^{out}_5) = V_{sat}(S_6) = V_{sat}(S_7) = V_{sat}(S_8) = 0$, and $V_{sat}(S_9) = 0.94$. Therefore, we remove state S_5 from both $\texttt{Acc}^{\times}$ and S_p, and further remove states S_1, S_6, S_7, S_8, S_9 from S_p, and thus obtain $S_p = \{S_2, S_3, S_4\}$ and $\texttt{Acc}^{\times} = \{S_4\}$ for the second iteration.

The product MDPST $\mathcal{M}^{\times}_{sub}$ for the second iteration is depicted in Fig. 5. By repeating the process in iteration 1, one can get that $V_{sat}(S_2) = V_{sat}(S_3) = V_{sat}(S^{in}_4) = V_{sat}(S^{out}_4) = 1$. Thus, one has that flag$= 0$. The iteration terminates. The WR is then given by $W^{\times} = \{S_2, S_3, S_4\}$.

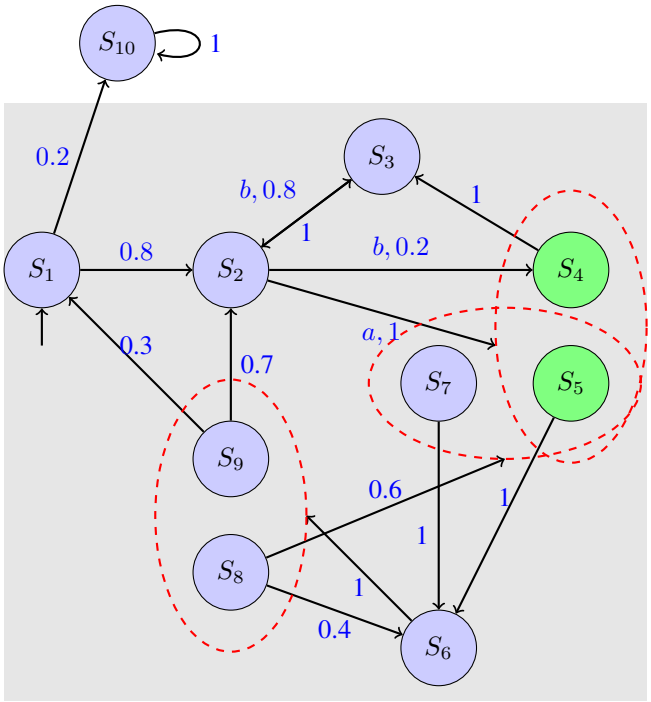

Fig. 3: The product MDPST $\mathcal{M}^{\times}$, where the sub-MDPST $\mathcal{M}^{\times}_{sub}$ (for the first iteration) is the part within the shaded box.

C. Proof of Theorem 1

We prove Theorem 1 in two steps.

a) Step 1: We show the correctness of Algorithm 1, i.e., that the output $W^{\times}$ of Algorithm 1 is indeed the WR of the product MDPST $\mathcal{M}^{\times}$. More specifically, for every state $s^{\times} \in W^{\times}$, we have that

- there exists a strategy $\sigma^{\times}$ from $s^{\times}$ such that $\Pr^{\sigma^{\times}(s^{\times})}_{\mathcal{M}^{\times}}(\square\Diamond \texttt{Acc}^{\times}) = 1$.

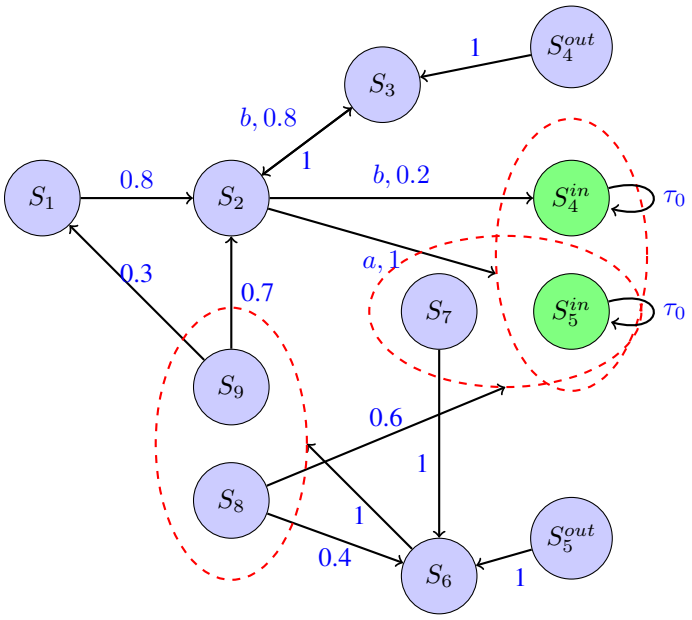

Fig. 4: The constructed product MDPST $\hat{\mathcal{M}}^{\times}_{sub}$.

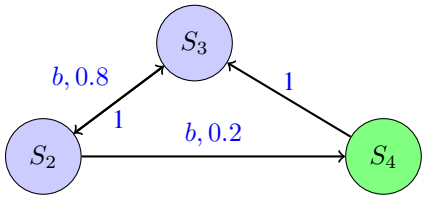

Fig. 5: The product MDPST $\mathcal{M}^{\times}_{sub}$ for the second iteration.

First, for a state $s^{\times} \in W^{\times}$, $\Pr^{\sigma^{\times}(s^{\times})}_{\mathcal{M}^{\times}}(\Diamond \texttt{Acc}^{\times}) = 1$, since $\hat{V}_{sat}(s^{\times}) = 1$, which means that there exists a strategy starting from $s^{\times}$ and leading to the virtual copies of accepting states in I_{in} with probability one. If $s \in W^{\times}$, there must be the virtual copies of accepting states in I_{out} that belong to $W^{\times}$, otherwise those states $s^{out} \in I_{out}$ will be removed from $\texttt{Acc}^{\times}$ (cf. line 10). Since $s \in W^{\times}$, we have a strategy σ_1 to generate a run from s reaching an accepting state, say $s_1^{in} \in I_{in}$, against any nature γ. Then we know that the virtual copy $s_1^{out} \in I_{out}$ also belongs to $W^{\times}$, otherwise the accepting state s_1 will be removed from $\texttt{Acc}^{\times}$ (cf. line 10). As a result, there exists a strategy σ_2 to generate a run from s_1^{out} (and thus s_1) to an accepting state $s_2^{in} \in I_{in}$ against any nature γ with probability one. Similarly, from s_2^{out}, we can extend the run to another accepting state s_3^{in}, and so on. Since the number of accepting states in $W^{\times}$ is finite, there must be an accepting state that is visited infinitely often. Hence, by merging the virtual copies of accepting states into their original state in this generated run, we can obtain an accepting run in $\mathcal{M}^{\times}$. Moreover, the probability measure of this run is 1, as each constructed fragment run has probability measure 1. By combining all these strategies $\sigma_1, \sigma_2, \cdots, \sigma_n$ constructed up to the point where the nature γ has repeated its behavior as it only has finite memory, we can obtain the strategy $\sigma = \sigma_1 \cdot \sigma_2 \cdots \sigma_n$. It immediately follows that $\Pr^{\sigma^{\times}(s^{\times})}_{\mathcal{M}^{\times}}(\Box\Diamond \texttt{Acc}^{\times}) = 1$.

As long as a run gets trapped in the WR, we have a strategy to keep the run in the WR with probability one. It immediately follows that:

$$\max_{\sigma^{\times} \in \Pi_{\mathcal{M}^{\times}}} \{\Pr^{\sigma^{\times}}_{\mathcal{M}^{\times}}(\Diamond\Box \mathrm{W}^{\times})\} = \max_{\sigma^{\times} \in \Pi_{\mathcal{M}^{\times}}} \{\Pr^{\sigma^{\times}}_{\mathcal{M}^{\times}}(\Diamond \mathrm{W}^{\times})\} = \max_{\sigma^{\times} \in \Pi_{\mathcal{M}^{\times}}} \{\Pr^{\sigma^{\times}}_{\mathcal{M}^{\times}}(\Diamond\Box \texttt{Acc}^{\times})\}.$$

b) Step 2: We show that

$$\max_{\sigma \in \Pi_{\mathcal{M}}} \{\Pr^{\sigma}_{\mathcal{M}}(\varphi)\} = \max_{\sigma^{\times} \in \Pi_{\mathcal{M}^{\times}}} \{\Pr^{\sigma^{\times}}_{\mathcal{M}^{\times}}(\Diamond\Box \mathrm{W}^{\times})\}$$

holds by verifying both sides of the inequality and subsequently constructing the induced policy on $\mathcal{M}$.

Let $p = \max_{\sigma}\{\Pr^{\sigma}_{\mathcal{M}}(\varphi)\}$. In other words, there is an *optimal* robust strategy σ such that, for every possible nature γ of $\mathcal{M}$, we have $\Pr^{\sigma,\gamma}_{\mathcal{M}}(\varphi) \geq p$ and there exists some nature γ' such that $\Pr^{\sigma,\gamma'}_{\mathcal{M}}(\varphi) = p$. For every fixed strategy σ and nature γ, we can obtain a Markov chain $\mathcal{M}_{\sigma,\gamma}$.

To prove the direction of $\leq$, we will construct a strategy $\sigma^{\times}$ for $\mathcal{M}^{\times}$ such that, for every nature $\gamma^{\times}$, $\Pr^{\sigma^{\times},\gamma^{\times}}_{\mathcal{M}^{\times}}(\Diamond\Box \mathrm{W}^{\times}) \geq p$. Observe that every nature $\gamma^{\times}$ for $\mathcal{M}^{\times}$ induces a nature γ for $\mathcal{M}$ by ignoring the component from the automaton $\mathcal{A}$. Both Markov chains $\mathcal{M}_{\sigma,\gamma}$ and $\mathcal{M}^{\times}_{\sigma^{\times},\gamma^{\times}}$ (ignoring the acceptance condition) will have the same probability to generate traces that satisfy φ, because the probability in $\mathcal{M}^{\times}$ comes only from $\mathcal{M}$. Thus, for every nature $\gamma^{\times}$ we construct $\sigma^{\times}$ by following σ of $\mathcal{M}$ within the state space $S \times Q_i$ and $S \times Q_{acc}$, where Q_i (respectively, Q_{acc}) is the deterministic part of Q before (respectively, after) seeing an ϵ-transition. This is possible because the automaton transitions within Q_i and Q_{acc} separately are deterministic. To resolve the nondeterministic jump from Q_i to Q_{acc} (i.e., ϵ-transitions), we select the automaton successors according to the behavior of the Markov chain $\mathcal{M}_{\sigma,\gamma}$. That is, before $\mathcal{M}_{\sigma,\gamma}$ enters a bottom strongly connected component (BSCC), $\sigma^{\times}$ follows σ in $\mathcal{M}$ and stays within the Q_i part in $\mathcal{A}$. The moment when $\mathcal{M}_{\sigma,\gamma}$ enters a BSCC from a product state (s, q), according to [34], there is a way to select a successor q' in Q_{acc} for the state q such that the trace generated by $\mathcal{M}_{\sigma,\gamma}$ from (s, q) satisfies φ if, and only if, the corresponding run from state q' is accepted by $\mathcal{A}$. So, from state (s, q), $\sigma^{\times}$ selects the action $\epsilon_{q'}$ and the run moves to (s, q'). Afterwards, $\sigma^{\times}$ follows σ in $\mathcal{M}$ and the run in $\mathcal{A}$ is again deterministic. We can see that $\sigma^{\times}$ basically imitates the behavior of σ except for resolving the nondeterminism in the automaton $\mathcal{A}$. One can see that every trace ρ in $\mathcal{M}_{\sigma,\gamma}$ corresponds to a trace $\rho^{\times}$ in $\mathcal{M}^{\times}_{\sigma^{\times},\gamma^{\times}}$ with equal transition probabilities except for the ϵ-transition, which has probability 1. Moreover, as aforementioned, if ρ satisfies φ, the run $\rho^{\times}$, projected to the second component, is also an accepting run in $\mathcal{A}$.

Now we only need to prove that the accepting trace $\rho^{\times}$ will stay within the WR $W^{\times}$ in the end. Obviously, all states that occur in $\rho^{\times}$ infinitely many times, denoted C, can reach each other. It must be the case that $C \subseteq W^{\times}$. This

is because the run $\rho^\times$ already enters a BSCC of $\mathcal{M}_{\sigma,\gamma}$ and the probability measure of this run in the BSCC is 1. Since ρ is accepting, there must be an accepting state $s^\times \in \mathtt{Acc}^\times$ that can visit itself with probability one under the strategy σ against any adversarial nature γ. Moreover, $s^\times$ clearly can be reached from the initial state. Therefore, after applying Algorithm 1, $s^\times$ must belong to $W^\times$. Since all states in the BSCC C can reach accepting state $s^\times$ with probability one against any nature γ, we have that $C \subseteq W^\times$. That is, the accepting run $\rho^\times$ eventually stays in $W^\times$. We then have that $\Pr_{\mathcal{M}^\times}^{\sigma^\times,\gamma^\times}(\Diamond\Box \mathrm{W}^\times) \geq \Pr_{\mathcal{M}}^{\sigma,\gamma}(\varphi) \geq p$ as σ is an optimal robust strategy for $\mathcal{M}$.

The direction of $\geq$ is trivial. For any run that eventually stays within the WR $W^\times$, it will almost surely visit an accepting state infinitely many times. Hence, the run must meet the Büchi condition. It is clear that any strategy $\sigma^\times$ on $\mathcal{M}^\times$ can induce a policy for $\mathcal{M}$ by eliminating the ϵ-transitions. Therefore, any path following $\sigma^\times$ that meets the Büchi condition will induce a path of $\mathcal{M}$ that is accepting by $\mathcal{A}$ induced from φ, where the non-determinism of $\mathcal{A}$ is resolved by ϵ-transitions of $\sigma^\times$, thus satisfying φ.

Therefore, we have completed the proof.