Automata-based Optimal Planning with Relaxed Specifications

Disha Kamale, Eleni Karyofylli, and Cristian-Ioan Vasile

***Abstract*— In this paper, we introduce an automata-based framework for planning with relaxed specifications. User relaxation preferences are represented as *weighted finite state edit systems* that capture permissible operations on the specification, substitution and deletion of tasks, with complex constraints on ordering and grouping. We propose a *three-way product automaton* construction method that allows us to compute minimal relaxation policies for the robots using shortest path algorithms. The three-way product automaton captures the robot's motion, specification satisfaction, and available relaxations at the same time. Additionally, we consider a bi-objective problem that balances temporal relaxation of deadlines within specifications with changing and deleting tasks. Finally, we present the runtime performance and a case study that highlights different modalities of our framework.**

I. INTRODUCTION

Robots are increasingly required to perform complex tasks with rich temporal and logical structure. In recent years, automata-based approaches have been widely used for solving robotic path planning problems wherein an automaton is constructed from mission specifications posed as temporal logic formulae (e.g. LTL, scLTL, STL, TWTL) [1]–[4]. Using shortest path algorithms on the product models between abstract robot motion models and specification automata, optimal satisfying trajectories are synthesized.

This traditional approach, albeit useful, does not consider modifying mission specifications in case satisfaction is infeasible. This implies that even the sub-parts of the specification that might be feasible will not be executed. In many real-world scenarios, it is often preferable that the robot performs at least some part of the assigned task even if it cannot be satisfied in its entirety. Consider the following mission specification for data collection: *"Collect data from region G_1 and then region G_2 and then upload it at U_1. Collect data from G_3 and upload it at U_2. Always avoid obstacles."* In case an obstacle makes it impossible to reach U_2, it is still preferred to receive the data from G_1 and G_2. Thus, we need to consider relaxed satisfaction semantics to handle infeasible mission specifications.

In literature, the problem of specification relaxation has been formulated in various ways. *Minimum violation* is considered in [5]–[7] for self-driving cars, where policies are computed with minimal rules of the road violation based on priorities. Their approach is based on the removal of violating symbols from the input of the specification automata to produce satisfying runs. In [8], *minimum revision* of tasks for office robots is explored. Their approach allows changing of tasks based on user-provided substitution costs. A similar problem is tackled in [9], but for infinite horizon case with Büchi automata. Both works modify the input stream of the specification automaton to induce feasibility. *Partial satisfaction* [10], [11] approaches aim to compute policies that minimize distance to satisfaction given by paths to accepting states in specification automata. In a different direction, [4], [6], [12] consider *temporal relaxation* of deadlines to complete missions. Their approach introduces annotated automata that capture all deadline relaxations from specifications, to compute policies with minimal delays. Some of these works combine relaxation of specifications with maximizing satisfaction probability [8], [10], [13]. All these works use automata-based techniques. However, all have specialized approaches that can not be readily combined. Moreover, they operate on a symbol-by-symbol basis rather than words translations that capture rich relaxation preferences on groups of tasks.

This work proposes a framework that brings together the core notions of several automata-based methods for planning with relaxation and allows for handling complex specifications and relaxation preferences in symbolic path planning. The main contributions of the paper are: (1) the formulation of the *minimum relaxation problem* that unifies several existing problems and generalizes them to relaxation rules with memory; (2) an automata-based formalism to capture user relaxation preferences via WFSEs; (3) an automata-based planning framework that uses a novel three-way product automaton construction that simultaneously captures the motion, specification, and relaxation preference models; and (4) case studies that demonstrate different instances of specification relaxation, a bi-objective optimal synthesis problem, and the runtime performance. To the best of our knowledge, this is the first time relaxation rules are considered that account for complex ordering and grouping of sub-tasks when revising mission specifications.

II. PRELIMINARIES

In this section, we introduce notation used throughout the paper, and briefly review the main concepts from formal languages, automata theory, and formal verification. For a detailed exposition of these topics, we refer the reader to [14], [15] and the references therein.

We denote the range of integer numbers as $[[a, b]] = \{a, \dots, b\}$, and $[[a]] = [[0, a]]$. Let Σ be a finite set. We denote the cardinality and the power set of Σ by $|\Sigma|$ and 2^{Σ}, respectively. A *word* over Σ is a finite or infinite sequence of elements from Σ. In this context, Σ is also called an *alphabet*. The length of a word w is denoted by $|w|$. Let k, $i \leq j$ be non-negative integers. The k-th element of w is denoted by w_k, and the *sub-word* $w_i, \dots, w_j$ is denoted by $w_{i,j}$. Let $I = \{i_0, i_1 \dots\} \subseteq [[|w|]]$. The *sub-sequence* $w_{i_0}, w_{i_1} \dots$ is denoted by w_I. A set of words over an alphabet Σ is called a *language* over Σ. The language of all finite words over Σ is denoted by Σ^*.

The authors are with Lehigh University, Bethlehem, PA 18015. {ddk320, elk222, cvasile}@lehigh.edu

Definition 2.1 (Deterministic Finite State Automaton): A deterministic finite state automaton (DFA) is a tuple $\mathcal{A} = (S_\mathcal{A}, s_0^\mathcal{A}, \Sigma, \delta_\mathcal{A}, F_\mathcal{A})$, where: $S_\mathcal{A}$ is a finite set of states; $s_0^\mathcal{A} \in S_\mathcal{A}$ is the initial state; Σ is the input alphabet; $\delta_\mathcal{A}\colon S_\mathcal{A} \times \Sigma \to S_\mathcal{A}$ is the transition function; $F_\mathcal{A} \subseteq S_\mathcal{A}$ is the set of accepting states.

A trajectory of the DFA $\mathbf{s} = s_0 s_1 \dots s_{n+1}$ is generated by a finite sequence of symbols $\boldsymbol{\sigma} = \sigma_0\sigma_1\dots\sigma_n$ if $s_0 = s_0^\mathcal{A}$ is the initial state of $\mathcal{A}$ and $s_{k+1} = \delta_\mathcal{A}(s_k, \sigma_k)$ for all $k \geq 0$. A finite input word $\boldsymbol{\sigma}$ over Σ is said to be accepted by a finite state automaton $\mathcal{A}$ if the trajectory of $\mathcal{A}$ generated by $\boldsymbol{\sigma}$ ends in a state belonging to the set of accepting states, i.e., $F_\mathcal{A}$. The *(accepted) language* of a DFA $\mathcal{A}$ is the set of accepted input words denoted by $\mathcal{L}(\mathcal{A})$.

Definition 2.2 (Transition System): A weighted transition system (TS) is a tuple $\mathcal{T} = (X, x_0^\mathcal{T}, \delta_\mathcal{T}, AP, h, w_\mathcal{T})$, where: X is a finite set of states; $x_0^\mathcal{T} \in X$ is the initial state; $\delta_\mathcal{T} \subseteq X \times X$ is a set of transitions; AP is a set of properties (atomic propositions); $h\colon X \to 2^{AP}$ is a labeling function; $w_\mathcal{T}\colon \delta_\mathcal{T} \to \mathbb{Z}_{>0}$ is a weight function.

A *trajectory* (or run) of the system is an infinite sequence of states $\mathbf{x} = x_0 x_1 \dots$ such that $(x_k, x_{k+1}) \in \delta_\mathcal{T}$ for all $k \geq 0$, and $x_0 = x_0^\mathcal{T}$. The set of all trajectories of $\mathcal{T}$ is $Runs(\mathcal{T})$. A state trajectory $\mathbf{x}$ generates an *output trajectory* $\mathbf{o} = o_0 o_1 \dots$, where $o_k = h(x_k)$ for all $k \geq 0$. We also denote an output trajectory by $\mathbf{o} = h(\mathbf{x})$. The *(generated) language* corresponding to a TS $\mathcal{T}$ is the set of all generated output words, which we denote by $\mathcal{L}(\mathcal{T})$. We define the weight of a trajectory as $w_\mathcal{T}(\mathbf{x}) = \sum_{k=1}^{|\mathbf{x}|} w_\mathcal{T}(x_{k-1}, x_k)$.

III. BACKGROUND ON PLANNING WITH RELAXED SPECIFICATIONS

In this section, we review existing temporal logic-based planning problems that consider specification relaxation in case of infeasibility. In the subsequent sections, we generalize all these problems, and propose an automata-based framework amenable to off-the-shelf synthesis methods instead of customized solutions. For cohesiveness and clarity, we present the core features of the relaxed TL planning problems, in some cases, adapted to finite-time.

Throughout the paper, we assume that the motion of a robot is captured by a finite weighted transition system $\mathcal{T}$ and the finite-time specifications expressed using temporal logics (TL), e.g. scLTL [16], [17], TWTL [18], BLTL [19], and Finite LTL [20], and regular expressions (RE) [15], [21]. All of these representations can be translated to DFAs using off-the-shelf tools e.g., *spot* [17], *scheck* [16], *pytwtl* [18]. Thus, we consider specifications given as a DFA $\mathcal{A}$.

1. Canonical Problem (CP):

Problem 3.1: Find a trajectory for $\mathcal{T}$ such that the output trajectory is accepted by $\mathcal{A}$.

Optimality: Minimize the weight of the trajectory.

In the canonical problem, no relaxations are permitted.

2. Minimum Violation Problem (MVP): Let $\mathbf{o}$ be a word over 2^{AP}, and ϖ per symbol violation cost. The *violation cost* of $\mathbf{o}$ with respect to $\mathcal{A}$ is $\min_{I \subseteq [[|w|]]} \varpi\,|I|$ s.t. $\mathbf{o}_{[[|w|]]\setminus I} \in \mathcal{L}(\mathcal{A})$. The violation cost of a TS trajectory $\mathbf{x}$ is induced by the output word $\mathbf{o} = h(\mathbf{x})$.

Problem 3.2 (Minimum violation): Find a trajectory for $\mathcal{T}$ such that a sub-sequence of the output trajectory is accepted by $\mathcal{A}$.

Optimality: Minimize the violation cost of the trajectory.

3. Minimum Revision Problem (MRP): Let $\mathbf{o}$ be a word over 2^{AP}, and $c\colon 2^{AP} \times 2^{AP} \to \mathbb{R}$ the symbol substitution cost. The *revision cost* of $\mathbf{o}$ with respect to $\mathcal{A}$ is $\min_{\mathbf{o}' \in \mathcal{L}(\mathcal{A})} \sum_{i=0}^{|\mathbf{o}|} c(\mathbf{o}_i, \mathbf{o}'_i)$ s.t. $|\mathbf{o}'| = |\mathbf{o}|$, where $\mathbf{o}'$ is the *revised word*.

The symbol substitution cost function c is defined such that there is no penalty for no substitution, i.e., $c(\sigma, \sigma) = 0$ for all $\sigma \in 2^{AP}$.

Problem 3.3 (Minimum revision): Let c be the symbol substitution cost. Find a trajectory for $\mathcal{T}$ such that a revision of the output trajectory is accepted by $\mathcal{A}$.

Optimality: Minimize the revision cost of the trajectory.

For MRP, as our framework operates on groups of symbols (words), we refer to it as Minimum Word Revision Problem (MWRP) throughout the paper.

4. Hard-Soft Constraints Problem (HSC):

Problem 3.4: Let $\mathcal{A}_H$ and $\mathcal{A}_S$ be two specification DFAs. Find a trajectory for $\mathcal{T}$ such that the output trajectory is accepted by $\mathcal{A}_H$, and, if possible, by $\mathcal{A}_S$.

Optimality: Minimize the cost of the trajectory.

We adapt the HSC problem from [2] for finite-time specifications, where we replace Büchi automata with DFAs.

5. Partial Satisfaction (PS): Let $\mathbf{o} \in 2^{AP}$. The *continuation cost* of $\mathbf{o}$ with respect to $\mathcal{A}$ is $\min_{\mathbf{o}^c \in (2^{AP})^*} |\mathbf{o}^c|$ s.t. $\mathbf{o}' = \mathbf{o}\mathbf{o}^c \in \mathcal{L}(\mathcal{A})$, where $\mathbf{o}'$ is a *continuation* of $\mathbf{o}$.

Problem 3.5: Find a trajectory for $\mathcal{T}$ such that a continuation of the output trajectory is accepted by $\mathcal{A}$.

Optimality: Minimize the cost of the continuation.

The problem minimizes the amount of work still needed to satisfy the specification from partial trajectories.

6. Temporal Relaxation (TR):

We leverage Time window temporal logic (TWTL) [1], a specification language for robotics applications with explicit time bounds. We refer the reader to [18] for a detailed exposition of TWTL syntax and semantics.

Let ϕ be a TWTL formula and $\boldsymbol{\tau} \in \mathbb{Z}^m$, where m is the number of *within* operators contained in ϕ. The *τ-relaxation* of ϕ is a TWTL formula $\phi(\boldsymbol{\tau})$, where each subformula of the form $[\phi_i]^{[a_i,b_i]}$ is replaced by $[\phi_i]^{[a_i,b_i+\tau_i]}$. We consider the linear temporal relaxation (LTR), where the LTR of $\phi(\boldsymbol{\tau})$ is $|\boldsymbol{\tau}|_{LR} = \sum_j \tau_j$, and $\phi(\boldsymbol{\tau})$ is a $\boldsymbol{\tau}$-relaxation of ϕ.

Problem 3.6: Find a trajectory for $\mathcal{T}$ such that the output trajectory satisfies the relaxed formula $\phi(\boldsymbol{\tau})$ for some relaxation $\boldsymbol{\tau}$ of the deadlines in ϕ.

Optimality: Minimize the linear temporal relaxation.

7. Planning: All the problems above are solved by constructing a standard product automaton between the motion model $\mathcal{T}$ and the specification DFA $\mathcal{A}$. Planning with relaxed semantics is achieved via custom pre-processing procedures of $\mathcal{A}$, and custom shortest path algorithms. In the following, we show that all these problems can be captured via an additional automata-based model for *user task relaxation*, and solved using standard shortest path methods applied on a novel 3-way product. Moreover, MVP and MRP are restricted to relaxations of a single symbol at a time. Our framework can handle rich relaxation rules that involve changing groups of symbols (words).

IV. PROBLEM FORMULATION

In this section, we introduce an optimal planning problem for finite system abstractions with temporal logic goals. We define a cost function based on user preferences on task relaxation in case satisfying the given specification is infeasible. Using the *user task preference* we define an optimal planning problem over the finite motion model, where the specification language is enlarged to ensure feasibility with appropriate penalties.

Definition 4.1 (User Task Preference): Let L be a language over the alphabet 2^{AP}. A *user task preference* is a pair (R, w_R), where $R \subseteq L \times (2^{AP})^*$ is a relation that captures how words in L can be transformed to words from $(2^{AP})^*$, and $w_R \colon R \to \mathbb{R}$ represents the cost of the word transformations. The relation R can also be understood as a multi-valued function $R \colon L \rightrightarrows (2^{AP})^*$.

Problem 4.1 (Minimum Relaxation): Given a transition system $\mathcal{T}$, a specification DFA $\mathcal{A}$, and a task relaxation preference (R, w_R), find a trajectory $\mathbf{x} \in Runs(\mathcal{T})$ that minimizes the task cost. Formally, we have $\min_{\mathbf{x} \in Runs(\mathcal{T})} w_R(\mathbf{o}, \mathbf{o}')$s.t. $\mathbf{o} \in \mathcal{L}(\mathcal{A}), \mathbf{o}' = h(\mathbf{x}) \in \mathcal{L}(\mathcal{T}) \cap R(\mathbf{o})$ where $R \colon \mathcal{L}(\mathcal{A}) \rightrightarrows (2^{AP})^*$ and $w_R \colon R \to \mathbb{R}$.

Task preferences can be used to substitute and delete tasks which are associated with words. These generalize edit-space operations on single symbols to words, and the optimization problem Problem 4.1 generalizes the Levenstein distance between languages of finite words.

User task preferences (R, w_R) can be represented in many ways. We consider the user preferences for relaxation provided as regular expressions (RE) and regular grammars that can be readily translated to automata using standard methods [15]. Consider the following example.

Example 4.1: Specification: visit region P1 for 1 time unit followed by P2 for 2 time units. Should visiting either or both be not possible, the substitution rules are: Substitute the visit to P1 by visiting Q1 for 2 time units with a penalty of d1, and the visit to P2 by visiting S1 for 2 time units followed by S2 for 1 time unit with a penalty of d2. Formally, $R = ((\bullet/\bullet, 0)^*(\{Q1\}/\{P1\}, 0)(\{Q1\}/\epsilon, d1)(\{S1\}/\{P2\}, 0)$ $(\{S1\}/\{P2\}, 0)$ $(\{S2\}/\epsilon, d2)(\bullet/\bullet, 0)^*$, where $\bullet$ represents any symbol in 2^{AP}, ϵ denotes a deleted symbol, $\{Q1\}/\{P1\}$ denotes that $\{P1\}$ is substituted by $\{Q1\}$, and d1, d2 denote the penalties for the corresponding substitutions. Note that the transformation can be performed multiple times due to the outer Kleene star operator. Alternatively, the transformation rules are $P1 \mapsto^{d1} Q1Q1$ and $P2P2 \mapsto^{d2} S1S1S2$. and the possible alternatives are: a) $r'_1 = Q1Q1P2P2$, b) $r'_2 = P1S1S1S2$, and c) $r'_3 = Q1Q1S1S1S2$.

We use weight computation functions $f_w(\cdot)$ that combine TS and WFSE weights to capture multiple semantics for relaxation penalties. Next, we show that Prob. 4.1 captures all problems from Sec. III. The proof is provided in [22].

Proposition 4.1: Problems 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6 are instances of Prob. 4.1

V. UNIFIED AUTOMATA-BASED FRAMEWORK

In this section, we introduce a unified automata-based framework to capture user preference specifications, and to synthesize minimal relaxation policies.

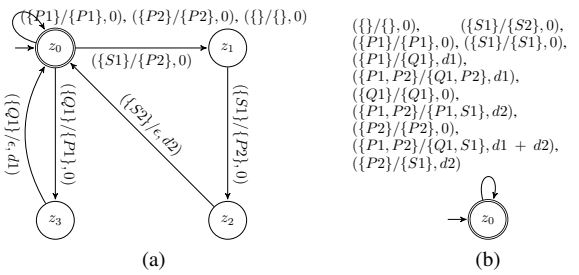


Fig. 1: (a) WFSE for word-word translations, (b) WFSE corresponding to symbol-symbol translations

A. Relaxation Specification

We consider two classes of problems related to task changes and deadline relaxations.

1) Task Relaxation: In this problem class, we allow parts of the specification to be substituted and/or removed. Preferences can be given in many formats, e.g., regular expressions and grammars, (Ex. 4.1). We introduce weighted finite state edit systems to represent user task relaxation preferences with bounded memory, where weights capture translation penalties.

Definition 5.1 (Weighted Finite State Edit System): A weighted finite state edit system (WFSE) is a weighted DFA $\mathcal{E} = (Z_\mathcal{E}, z_0^\mathcal{E}, \Sigma_\mathcal{E}, \delta_\mathcal{E}, F_\mathcal{E}, w_\mathcal{E})$, where $\Sigma_\mathcal{E} = \left(2^{AP} \cup \{\epsilon\}\right) \times \left(2^{AP} \cup \{\epsilon\}\right) \setminus \{(\epsilon, \epsilon)\}$, ϵ denotes a missing or deleted symbol, and $w_\mathcal{E} \colon \delta_\mathcal{E} \to \mathbb{R}$ is the transition weight function.

The alphabet $\Sigma_\mathcal{E}$ captures word edit operations (addition, substitution, or deletion of symbols). A transition $z' = \delta_\mathcal{E}(z, (\sigma, \sigma'))$ has input, output symbols σ and σ'. Given a word $\vec{\boldsymbol{\sigma}} = (\sigma_0, \sigma'_0)(\sigma_1, \sigma'_1) \ldots (\sigma_r, \sigma'_r) \in \mathcal{L}(\mathcal{E})$, $r = |\vec{\boldsymbol{\sigma}}| - 1$, we call $\boldsymbol{\sigma} = \sigma_{i_0}\sigma_{i_1} \ldots \sigma_{i_n} \in (2^{AP})^*$ and $\boldsymbol{\sigma}' = \sigma'_{j_0}\sigma'_{j_1} \ldots \sigma'_{j_m} \in (2^{AP})^*$ obtained by removing only the symbol ϵ, the input and output words, where $m, n, r \in \mathbb{Z}_{>0}$, $m, n < r$, $0 \leq i_0 < \ldots < i_m < r$ and $0 \leq j_0 < \ldots < j_n < r$. Moreover, we say that $\mathcal{E}$ transforms $\boldsymbol{\sigma}$ into $\boldsymbol{\sigma}'$.

Note that WFSE is a special type of finite state transducer where the input and output alphabets are the same, and the empty symbol ϵ cannot be mapped to itself. Moreover, the weights capture translation penalties and can depend on the states and symbol translation pairs. We can use standard methods [15] to translate REs that express relaxation rules into WFSEs.

2) Temporal Relaxation: Temporal relaxation allows delays with respect to deadlines in the satisfaction of specifications. In the following, we consider annotated automata $\mathcal{A}_\infty$ computed from TWTL formulae [18] that capture all possible deadline relaxations. Formally, given TWTL formula ϕ, an annotated DFA $\mathcal{A}_\infty$ is a DFA such that $\mathcal{L}(\mathcal{A}_\infty) = \mathcal{L}(\phi(\infty))$, where $\phi(\infty)$ is satisfied by a word $\mathbf{o}$ if and only if $\exists \boldsymbol{\tau}' < \infty$ s.t. $\mathbf{o} \models \phi(\boldsymbol{\tau}')$. When the transition weights of the TS $\mathcal{T}$ represent (integer) durations, we construct an extended TS $\widehat{\mathcal{T}}$ from $\mathcal{T}$ such that all transitions have unit weight (duration). This additional step ensures that transitions of $\mathcal{A}_\infty$ and $\mathcal{T}$ are synchronized. See [18] for details.

B. Product Automaton Construction

The optimal control policy that takes into account the user preferences is computed based on a product automaton

between three models: (a) the robot motion model $\mathcal{T}$; (b) the user preferences WFSE $\mathcal{E}$; and (c) the specification DFA $\mathcal{A}$.

Definition 5.2 (Three-way Product Automaton): Given a TS $\mathcal{T} = (X, x_0^{\mathcal{T}}, \delta_{\mathcal{T}}, AP, h, w_{\mathcal{T}})$, a WFSE system $\mathcal{E} = (Z_{\mathcal{E}}, z_0^{\mathcal{E}}, \Sigma_{\mathcal{E}}, \delta_{\mathcal{E}}, F_{\mathcal{E}}, w_{\mathcal{E}})$, and a specification DFA $\mathcal{A} = (S_{\mathcal{A}}, s_0^{\mathcal{A}}, 2^{AP}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$, the product automaton is a tuple $\mathcal{P}_{\mathcal{E}} = (Q^{\mathcal{E}}, q_0^{\mathcal{E}}, \delta_{\mathcal{P}}^{\mathcal{E}}, F_{\mathcal{P}}^{\mathcal{E}}, w_{\mathcal{P}}^{\mathcal{E}})$ also denoted by $\mathcal{P}_{\mathcal{E}} = \mathcal{T} \times_{\mathcal{E}} A$, where: $Q^{\mathcal{E}} = X \times Z_{\mathcal{E}} \times S_{\mathcal{A}} \cup \{q_0^{\mathcal{E}}\}$ is the state space; $q_0^{\mathcal{E}} = (\triangleright, z_0, s_0)$ is the initial state; $\delta_{\mathcal{P}}^{\mathcal{E}} \subseteq Q^{\mathcal{E}} \times Q^{\mathcal{E}}$ is the set of transitions; $F_{\mathcal{P}}^{\mathcal{E}} = X \times F_{\mathcal{E}} \times F_{\mathcal{A}} \subseteq Q^{\mathcal{E}}$ is the set of accepting states; $w_{\mathcal{P}}^{\mathcal{E}} \colon \delta_{\mathcal{P}}^{\mathcal{E}} \to \mathbb{R}$ is the transition weight function.

A transition $((x, z, s), (x', z', s')) \in \delta_{\mathcal{P}}^{\mathcal{E}}$ if $(x, x') \in \delta_{\mathcal{T}}$ or $x' = x_0$, $z' = \delta_{\mathcal{E}}(z, (\sigma, \sigma'))$, $s' = \delta_{\mathcal{A}}(s, \sigma')$, and $\sigma = h(x')$. Note that we introduce a virtual TS state $\triangleright$ connected to x_0 to avoid the definition of a set of initial states and associated start weights. State $\triangleright$ is only used to simplify notation and implementation, and does not correspond to an actual state of the robot. The weight function is $w_{\mathcal{P}}^{\mathcal{E}}((q, q')) = f_w(w_{\mathcal{T}}(x, x'), w_{\mathcal{E}}(z, z'))$, where $f_w(\cdot)$ is an arbitrary function, $q = (x, z, s)$, $q' = (x', z', s')$, and $w_{\mathcal{T}}(\triangleright, x_0^{\mathcal{T}}) = 1$ by convention. A trajectory of $\mathcal{P}_{\mathcal{E}}$ is said to be accepting only if it ends in a state that belongs to the set of final states $F_{\mathcal{P}}^{\mathcal{E}}$. The projection of the trajectory $\mathbf{q} = q_0^{\mathcal{E}} q_1 \dots q_n$ onto the TS $\mathcal{T}$ is $\mathbf{x} = x_0 x_1 \dots x_{n-1}$, where $q_0^{\mathcal{E}}$ is the initial state of $\mathcal{P}_{\mathcal{E}}$, and $q_k = (x_{k-1}, z_k, s_k)$, for all $k \in [[1, n]]$. Similar to [1], [18], we construct $\mathcal{P}_{\mathcal{E}}$ such that only states that are reachable from the initial state, and reach a final state.

Algorithm 1: Optimal Planning Algorithm – `Plan()`

Input: TS $\mathcal{T}$, TL specification ϕ (scLTL, TWTL, etc.), user task specification RE (R, w_R), product weight computation function f_w
Output: optimal policy for the TS $\mathcal{T}$

1 Translate ϕ to DFA $\mathcal{A}$ using off-the-shelf tools
2 Translate (R, w_R) to WFSE $\mathcal{E}$ using standard methods
3 Compute the three-way PA $\mathcal{P}_{\mathcal{E}} = \mathcal{T} \times_{\mathcal{E}} \mathcal{A}$
4 Compute shortest path $\mathbf{q}^*$ in $\mathcal{P}_{\mathcal{E}}$ from the initial state $q_0^{\mathcal{E}}$ to a final in $F_{\mathcal{P}}^{\mathcal{E}}$ using weights $w_{\mathcal{P}}^{\mathcal{E}}$
5 Project $\mathbf{q}^*$ onto $\mathcal{T}$ to obtain the optimal policy $\mathbf{x}^*$
6 **return** $\mathbf{x}^*$

C. Optimal Planning

The planning procedure `Plan()` for computing the optimal trajectory is outlined in Alg. 1. The weights $w_{\mathcal{P}}^{\mathcal{E}}$ of $\mathcal{P}_{\mathcal{E}}$ used for computing the shortest path depend on whether we wish to minimize task or deadline relaxation as shown next.

1) Task Cost: Let $\mathbf{q} = q_0 \dots q_n$ be a trajectory of $\mathcal{P}_{\mathcal{E}}$. The task cost of $\mathbf{q}$ is $C_{\mathcal{E}}(\mathbf{q}) = \sum_{k=0}^{n-1} c_{\mathcal{E}}(q_k, q_{k+1})$, $c_{\mathcal{E}}(q_k, q_{k+1}) = f_w(w_{\mathcal{T}}(x_k, x_{k+1}), w_{\mathcal{E}}(z_k, z_{k+1}))$, where $c_{\mathcal{E}}$ is the transition weight, $q_k = (x_k, z_k, s_k)$ for all $k \in [[0, n]]$. The task cost takes into account the penalties associated with substitution and deletion of tasks represented as sub-words of the TS's output words. The optimal trajectory is computed as $\texttt{Plan}(\mathcal{T}, \mathcal{A}, \mathcal{E}, c_{\mathcal{E}})$ using Alg. 1.

2) Temporal Relaxation Cost: In this case, the cost is captured by LTR introduced in Sec. III that aggregates all delays captured by the annotated specification DFA $\mathcal{A}_\infty$. The PA is denoted by $\mathcal{P}_{\mathcal{E}_0}$ and the optimal trajectory is computed as $\texttt{Plan}(\widehat{\mathcal{T}}, \mathcal{A}_\infty, \mathcal{E}_0, c_{TR})$, where $\widehat{\mathcal{T}}$ is the extended TS, and $\mathcal{E}_0$ is a trivial WFSE with a single node and a pass-through self-loop (leaves symbols unchanged and has weight 0). The temporal relaxation cost of $\mathbf{q}$ is $C_{TR}(\mathbf{q}) = |\mathbf{q}|, c_{TR}(q, q') = 1, \forall (q, q') \in \delta_{\mathcal{P}}^{\mathcal{E}_0}$, where $\mathbf{q}$ is a trajectory of $\mathcal{P}_{\mathcal{E}_0}$. Minimizing the length of trajectory $\mathbf{q}$ is equivalent to minimizing $|\tau|_{LR}$. This follows from the results in [18].

3) Bi-objective Cost: We consider cases where a robot can trade-off between changing tasks and delaying their satisfaction. The solution combines an annotated specification automaton $\mathcal{A}_\infty$ with a (non-trivial) relaxation preference WFSE $\mathcal{E}$ to compute policies in $\widehat{T}$ using $\texttt{Plan}(\widehat{\mathcal{T}}, \mathcal{A}_\infty, \mathcal{E}, c_{bi})$. The blended cost of a trajectory $\mathbf{q}$ is $\mathcal{C}_{bi}(\mathbf{q} \mid \lambda) = \lambda \mathcal{C}_{\mathcal{E}}(\mathbf{q}) + (1 - \lambda)\mathcal{C}_{TR}(\mathbf{q}), c_{bi}(q, q' \mid \lambda) = \lambda c_{\mathcal{E}}(q, q') + (1 - \lambda) c_{TR}(q, q')$, where the $\lambda \in [0, 1]$ is a parameter that trades-off between the two objectives, and c_{bi} is the bi-objective transition cost.

We compute the Pareto-optimal trajectories and the Pareto-front using a parametric Dijkstra's algorithm [23]. The Pareto-front for our bi-objective problem is composed of a finite number of isolated points in the $c_{TR} \times c_{\mathcal{E}}$ cost space.

D. Complexity Analysis

The construction of the three-way PA $\mathcal{P}_{\mathcal{E}}$, line 3 in Alg. 1, takes $\mathcal{O}(|\delta_{\mathcal{T}}| \times |\delta_{\mathcal{E}}| \times |\delta_{\mathcal{A}_\infty}|)$. Computing the shortest path $\mathbf{q}^*$ (line 4) is done with Dijkstra's algorithm which takes $\mathcal{O}(|\delta_{\mathcal{P}}^{\mathcal{E}}| + |Q^{\mathcal{E}}| \log |Q^{\mathcal{E}}|)$. Lastly, projection onto $\mathcal{T}$ (line 5) is linear in the size of $\mathbf{q}$. Crucially, our framework has the same asymptotic complexity as custom planning methods for the problems in Sec. III. MVP, MRP, and PS operate one symbol at a time, see the proof of Prop. 4.1 [22]. Their associated WFSEs have a single state with a self-loop, i.e., $|\delta_{\mathcal{E}}| = 1$ (see Fig.1b). Thus, the PA construction complexity degenerates to $\mathcal{O}(|\delta_{\mathcal{T}}| \times |\delta_{\mathcal{A}_\infty}|)$. For TR, the complexity also reduces since the WFSE has a single state; the deadline relaxation is captured by $\mathcal{A}_\infty$ [18]. Lastly, for HSC, we can choose $\mathcal{A}_H$ as specification DFA, and the WFSE can have same structure as $\mathcal{A}_S$ with penalty $M \gg 1$ if the soft constraint is not satisfied. For brevity, we omit the formal details. Thus, the complexity of PA construction becomes $\mathcal{O}(|\delta_{\mathcal{T}}| \times |\delta_{\mathcal{A}_H}| \times |\delta_{\mathcal{A}_S}|)$, the same as for custom methods (due to the quadratic complexity of language intersection [15]).

VI. CASE STUDIES

We consider a self-driving car in an urban environment, $\mathcal{T}$ abstraction of which is shown in Fig. 2. The robot has to visit task regions while avoiding obstacles. The green, red, and white states represent task regions, obstacles and waypoints respectively. The permissible directions of motion are represented by the edges. The global obstacle O and local obstacles $O2$, $O3$, if present, make it impossible to safely reach the neighbouring task regions. Even if $O3$ is absent, the robot can not stay at the $T1$ region next to $T2$ due to the no parking zone. The weights associated with transitions represent their duration. Transitions with no weight labels have unit weights. For all states except 14, 15, self transitions exist but haven't been included in the figure for simplicity. The transitions shown using yellow arrows and obstacle $O2$ are present only for problems 6-8, node 15 only for problem

8. We measure the time duration in minutes denoted by m for TWTL specifications.

For MWRP, HSC-MWRP, bi-objective problems, the relaxation preferences allow the substitution of $T1$ with $T2$, $T3$, $T4$, and $T5$ with costs 5, 8, 11, and 14, respectively. For MVP and HSC-MVP, the deletion cost is 10. For HSC, the cost for violation of the soft constraint is 10. The preferences and costs are captured by a WFSE with $|Z_{\mathcal{E}}| = 5$.

For MVP, MWRP, and HSC problems, we consider scenarios with O present (e.g., road construction, temporary closure), thereby making visits to $T1$ infeasible.

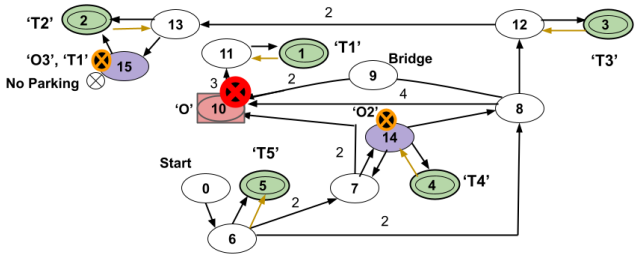


Fig. 2: Robot motion model

1) Canonical Problem (CP): The task specification is "Visit T1", i.e., $\phi = \mathbf{F}\ T1$, where $\mathbf{F}$ is the *eventually* operator. As obstacle O is absent, no relaxations are required. This case corresponds to a pass-through operation in $\mathcal{E}$. The optimal $\mathcal{T}$ trajectory is $(0, 6, 7, 10, 11, 1)$ which corresponds to the shortest path to $T1$ with a total cost $\mathcal{C}_{\mathcal{E}} = 9$.

2) Minimum Violation Problem (MVP): We consider the specification "Visit $T1$ and $T4$ while avoiding obstacles" that translates to the scLTL formula $\phi = \neg O\,\mathcal{U}\,T1 \wedge \neg O\,\mathcal{U}\,T4$. The optimal trajectory of $\mathcal{T}$ is $(0, 6, 7, 14, 4)$ as $T1$ is not reachable in the presence of O. The optimal cost is 15, including the cost of 10 for skipping $T1$.

3) Minimum Revision Problem (MWRP): In this case, the task specification is "Visit $T1$ while avoiding obstacles". If the task $T1$ is not feasible, revise the task according to the preferences given above. Here $\phi = \neg O\,\mathcal{U}\,T1$. The optimal $\mathcal{T}$ trajectory accepted by $\mathcal{A}$ is $(0, 6, 8, 12, 13, 2)$ with an optimal cost $\mathcal{C}_{\mathcal{E}}$ = 11, where $T1$ is substituted by $T2$ with cost 5.

4) Hard-Soft Constraints (HSC): This problem is implemented both in the presence and absence of obstacle O. The task specification is "Visit $T1$ while avoiding obstacles, and, if possible, take the $bridge$". The specification is $\phi = \phi_h \wedge \phi_s$, where $\phi_h = \neg O\,\mathcal{U}\,T1$ and $\phi_s = \mathbf{F}\,bridge$ are the hard and soft constraints, respectively. The cost of not satisfying ϕ_s is 10 and is added to the WFSE.

HSC-CP: Without obstacle O, the case is analogous to *CP* and, thus, the optimal $\mathcal{T}$ trajectory is $(0, 6, 8, 9, 10, 11, 1)$ with optimal cost $\mathcal{C}_{\mathcal{E}} = 10$.

HSC-MWRP: Here $T1$ is substituted by $T2$ which has the lowest substitution cost. Thus, the optimal $\mathcal{T}$ trajectory is $(0, 6, 8, 12, 13, 2)$ with cost $\mathcal{C}_{\mathcal{E}} = 20$ that includes the substitution cost 5 and the violation cost 10 for not going over the $bridge$.

HSC-MVP: $\phi_h = (\neg O\,\mathcal{U}\,T1) \wedge (\neg O\,\mathcal{U}\,T5)$, $\phi_s = \mathbf{F}\,bridge$. With obstacle O, only $T5$ can be visited. In the MVP case, the optimal $\mathcal{T}$ trajectory is $(0, 6, 7, 5)$ with an optimal cost of $\mathcal{C}_{\mathcal{E}} = 32$ that includes the costs of 10 for not visiting $T1$ and of 10 for not taking the $bridge$.

5) Temporal Relaxation (TR): In this example, the specification "Visit and stay in $T2$ for 2m within 0m to 6m." translates to a TWTL formula $\phi = [H^2 T2]^{[0,6]}$. As the minimum travel time to $T2$ from state 0 is 7m, the specification is relaxed to $\phi(\tau) = [H^2 T2]^{[0,6+\tau]}$ with $\tau = 3$ obtained by the optimal trajectory $(0, 6, 8, 12, 13, 2, 2, 2)$. The optimal cost is $\mathcal{C}_{TR} = 9$.

6) Multiple word-word translations: With $O2$ present, the task specification is: "Visit T4 for 2 consecutive instances and next, visit regions T4 and then T2 and next, visit regions T4 and T1. Avoid obstacles all the time." The corresponding scLTL specification is: $\neg O2\ U\ (\mathbf{F}(T4 \wedge X\ T4) \wedge XX\ ((\mathbf{F}\ T4 \wedge \mathbf{F}\ T2 \wedge (\neg T2\,\mathcal{U}\,T4)) \wedge X(\mathbf{F}\ T4 \wedge \mathbf{F}\ T1 \wedge (\neg O\,\mathcal{U}\,T1) \wedge (\neg T1\,\mathcal{U}\,T4))))$. If the task is infeasible, the substitution rules are: Substitute the first two instances of T4 (i.e. $\mathbf{F}T4 \wedge\ X\ T4$) by T5 with a total penalty of 6. Substitute the next occurrence of T4 by one T5 and two T3s with a total penalty of 4. Finally, delete the last occurrence of T4 with a penalty of 10 and substitute T1 by T2 with a penalty of 7. The trajectory obtained after relaxation is: (0,6,5,5,5,6,8,12,3,3,12,13,13,2,2) with a total cost $\mathcal{C}_{\mathcal{E}} = 35$.

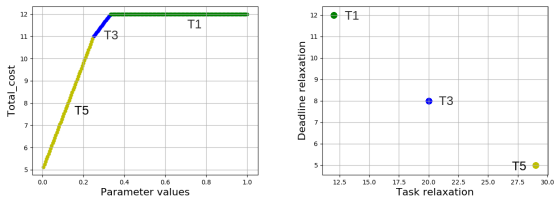
The above example demonstrates that our framework allows for multiple rules to be taken into account for different instances of the same symbol/word. Also, it highlights how the ordering is considered and retained during relaxation.

7) Bi-objective Problem: In the absence of O: "Visit $T1$ for 3m between 0m to 5m". If not feasible, use the substitution preferences. The DFA $\mathcal{A}_{\infty}$ is obtained from the TWTL formula $\phi = [H^3 T1]^{[0,5]}$. We obtain a set of Pareto-optimal trajectories and the corresponding intervals for parameter values. The intervals indicate the range of possible trade-offs between $C_{\mathcal{E}}$ and C_{TR} that correspond to the same Pareto-optimal trajectory. The set of solutions are: (1) $(0, 6, 5, 5, 5, 5)$, $\lambda \in [0, 0.25]$, corresponds to the minimum temporal relaxation $|\boldsymbol{\tau}|_{LR} = 0$ with $\mathcal{C}_{TR} = 5$ and $\mathcal{C}_{\mathcal{E}} = 29$; (2) $(0, 6, 8, 12, 3, 3, 3, 3)$, $\lambda \in [0.25, 0.33]$, strikes a balance between task and temporal relaxations with $\mathcal{C}_{\mathcal{E}} = 20$, $\mathcal{C}_{TR} = 8$, and $|\boldsymbol{\tau}|_{LR} = 3$; (3) $(0, 6, 8, 9, 10, 11, 1, 1, 1, 1)$, $\lambda \in [0.33, 1]$, achieves minimum task cost $\mathcal{C}_{\mathcal{E}} = 12$, $\mathcal{C}_{TR} = 12$ and $|\boldsymbol{\tau}|_{LR} = 7$. We now consider a word-word translation preference rule. Consider the specification "Visit T5 for 1s within first 3s from the start and immediately next, proceed to visit T4 for 2m within first 7m followed by T1 for 1m within first 5m of the mission. The local obstacle O2 should be avoided for the first 4m whereas the global obstacle O should be avoided for all 20m duration of the task." The corresponding TWTL formula is: "$H^{20}\neg O \wedge H^4 \neg O2 \wedge ([H^1 T5]^{[0,3]} \cdot [H^2 T4]^{[0,7]} \cdot [H^1 T1]^{[0,5]})$". The substitution rules are as follows: $T4 \mapsto^3 T5$, $T1 \mapsto^3 T5T3T2$, $T1 \mapsto^4 T3T2$, $T1 \mapsto^5 T3$. The trajectories obtained after relaxation are: 1) $(0, 6, 5, 5, 5, 5, 5, 5, 5, 6, 12, 3, 12, 13, 2)$, 2) $(0, 6, 5, 5, 6, 7, 14, 4, 4, 4, 14, 8, 12, 3, 3)$, 3) $(0, 6, 5, 5, 6, 7, 14, 4, 4, 4, 14, 8, 12, 3, 3, 13, 2)$.

8) Difference between symbol-symbol and word-word translations: Given that obstacle O is present and $O3$ is absent, the task is to visit T1 for 2m. As the route through node 15 is a no parking zone, there are no self-transitions on T1 at node 15. Given same substitution preferences as for MRP and if modelled as a wfse with a single state (see e.g., Fig. 1b) which is analogous to relaxations per-

User preference	Specification (ϕ)	O?	Optimal trajectory	cost
CP	$\mathbf{F}\,T1$	No	$\{0, 6, 7, 10, 11, 1\}$	9
MVP	$(\neg O\,\mathcal{U}\,T1) \wedge (\neg O\,\mathcal{U}\,T4)$	Yes	$\{0, 6, 7, 4\}$	14
MWRP	$\neg O\,\mathcal{U}\,T1$	Yes	$\{0, 6, 8, 12, 13, 2\}$	11
HSC-CP	$\phi_s = \mathbf{F}bridge$, $\phi_h = \neg O\,\mathcal{U}\,T1$	No	$\{0, 6, 8, 9, 10, 11, 1\}$	10
HSC- MWRP	—"—	Yes	$\{0, 6, 8, 12, 13, 2, \}$	20
HSC-MVP	$\phi_s = \mathbf{F}bridge$, $\phi_h = \neg O\,\mathcal{U}\,T1 \wedge \neg O\,\mathcal{U}\,T5$	Yes	$\{0, 6, 7, 5\}$	32
TR	$[H^2T2]^{[0,6]}$	Yes	$\{0, 6, 8, 12, 13, 2, 2, 2\}$	9

TABLE I: User specifications, preferences, and costs



(a) (b)

Fig. 3: (a): Total cost $\mathcal{C}_{bi}$ as a function of parameter λ, (b): Pareto front and the corresponding trajectories. (Note: The points on the Pareto front correspond to the respective colored line segments in 3a)

formed by the existing solutions, the shortest path obtained is (0,6,8,12,13,15,2) which violates the specification as it can pass through T1 (node 15) but not stay there. However, our framework with a wfse model similar to Fig.1a allows for this situation to be taken into account and the resultant trajectory is (0,6,8,12,13,2,2).

9) Runtime Performance: We study the effect of varying the sizes of $\mathcal{T}$ and $\mathcal{E}$ on the size and time taken for $\mathcal{P}_\mathcal{E}$ construction. This study was run on Dell Precision 3640 Intel i9-10900K with 64 GB RAM using python 2.7.12. We first keep the WFSE size constant at $\delta_\mathcal{E} = 8$ and vary the size of TS from $|X| = 118$ to $|X| = 300$ corresponding to which the $\mathcal{P}_\mathcal{E}$ size goes from $\left|Q^\mathcal{E}\right| = 352$ to $\left|Q^\mathcal{E}\right| = 898$. Note that $\mathcal{P}_\mathcal{E}$ construction retains only the reachable states. Whereas, the standard Cartesian product between $\mathcal{T}, \mathcal{E},$ and $\mathcal{A}$ varies from 1872 nodes to 4784 nodes. Similarly, when the TS is kept constant at $|X| = 22$ and the WFSE states are increased from $\delta_\mathcal{E} = 14$ to $\delta_\mathcal{E} = 453$, which corresponds to having n=453 substitution rules taken into account. For this, the $\mathcal{P}_\mathcal{E}$ size varies from $\left|Q^\mathcal{E}\right| = 106$ to $\left|Q^\mathcal{E}\right| = 986$ whereas the Cartesian product size increases from 616 nodes to 19932 nodes agreeing with the complexity analysis results from Section V-D.

VII. CONCLUSIONS

This work studies different existing approaches for specification relaxation and presents a unified and generalised automata-based framework that takes into account different instances of task and temporal relaxations and the trade-off between them. We propose the construction of a three-way product automaton that allows for word to word translations and temporal relaxations simultaneously. We demonstrate different instances of specification relaxation through case studies and show that the three-way product automaton construction scales linear in time with respect to transition system size. Future work includes extending the framework for control synthesis for stochastic systems while taking into account satisfaction probability.

REFERENCES

[1] C. I. Vasile and C. Belta, "An Automata-Theoretic Approach to the Vehicle Routing Problem," in *Robotics: Science and Systems Conference (RSS)*, Berkeley, California, USA, July 2014, pp. 1–9.

[2] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[3] J. Tumova, A. Marzinotto, D. V. Dimarogonas, and D. Kragic, "Maximally satisfying LTL action planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1503–1510.

[4] D. Aksaray, C. I. Vasile, and C. Belta, "Dynamic Routing of Energy-Aware Vehicles with Temporal Logic Constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016, pp. 3141–3146.

[5] J. Tumova, S. Karaman, C. Belta, and D. Rus, "Least-violating planning in road networks from temporal logic specifications," in *International Conference on Cyber-Physical Systems*, no. 17, Piscataway, NJ, USA, 2016, pp. 1–9.

[6] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1481–1488.

[7] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating Control Strategy Synthesis with Safety Rules," in *International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '13. New York, NY, USA: ACM, 2013, pp. 1–10.

[8] M. Lahijanian and M. Kwiatkowska, "Specification revision for markov decision processes with optimal trade-off," in *IEEE 55th Conference on Decision and Control*, Dec 2016, pp. 7411–7418.

[9] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the Minimal Revision Problem of Specification Automata," *International Journal Robotics Research*, vol. 34, no. 12, pp. 1515–1535, Oct. 2015.

[10] B. Lacerda, D. Parker, and N. Hawes, "Optimal Policy Generation for Partially Satisfiable Co-safe LTL Specifications," in *International Joint Conference on Artificial Intelligence*, 2015, pp. 1587–1593.

[11] 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 *AAAI Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 3664–3671.

[12] F. Penedo Álvarez, C. I. Vasile, and C. Belta, "Language-Guided Sampling-based Planning using Temporal Relaxation," in *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, USA, December 2016.

[13] 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.

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

[15] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[16] T. Latvala, "Efficient Model Checking of Safety Properties," in *10th International SPIN Workshop*, ser. Model Checking Software. Springer, 2003, pp. 74–88.

[17] A. Duret-Lutz, "Manipulating LTL formulas using Spot 1.0," in *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, ser. Lecture Notes in Computer Science, vol. 8172. Hanoi, Vietnam: Springer, Oct 2013, pp. 442–445.

[18] C.-I. Vasile, D. Aksaray, and C. Belta, "Time Window Temporal Logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, Aug 2017.

[19] I. Tkachev and A. Abate, "Formula-free Finite Abstractions for Linear Temporal Verification of Stochastic Hybrid Systems," in *International Conference on Hybrid Systems: Computation and Control*, Philadelphia, PA, April 2013, pp. 283–292.

[20] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *International Joint Conference on Artificial Intelligence*. AAAI Press, 2013, pp. 854–860.

[21] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal Approach to the Deployment of Distributed Robotic Teams," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 158–171, Feb 2012.

[22] D. Kamale, E. Karyofylli, and C.-I. Vasile, "Automata-based optimal planning with relaxed specifications," 2021.

[23] N. E. Young, R. E. Tarjan, and J. B. Orlin, "Faster parametric shortest path and minimum-balance algorithms," *Networks*, vol. 21, no. 2, pp. 205–221, 1991.