2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Reactive Temporal Logic Planning for Multiple Robots in Unknown Environments

Yiannis Kantaros, Matthew Malencia, Vijay Kumar, and George J. Pappas

***Abstract*—This paper proposes a new reactive mission planning algorithm for multiple robots that operate in unknown environments. The robots are equipped with individual sensors that allow them to collectively learn and continuously update a map of the unknown environment. The goal of the robots is to accomplish complex tasks, captured by global co-safe Linear Temporal Logic (LTL) formulas. The majority of existing temporal logic planning approaches rely on discrete abstractions of the robot dynamics operating in *known* environments and, as a result, they cannot be applied to the more realistic scenarios where the environment is initially unknown. In this paper, we address this novel challenge by proposing the first *reactive,* and *abstraction-free* LTL planning algorithm that can be applied for complex mission planning of multiple robots operating in unknown environments. Our algorithm is *reactive* in the sense that temporal logic planning is adapting to the updated map of the environment and *abstraction-free* as it does not rely on designing abstractions of robot dynamics. Our proposed algorithm is complete under mild assumptions on the structure of the environment and the sensor models. Our paper provides extensive numerical simulations and hardware experiments that illustrate the theoretical analysis and show that the proposed algorithm can address complex planning tasks in unknown environments.**

I. Introduction

Temporal logic motion planning has emerged as one of the main approaches for specifying a richer class of robot tasks than the classical point-to-point navigation and can capture temporal and Boolean requirements, such as sequencing, surveillance or coverage [1]–[11]. Common in these works is that they typically assume robots with *known* dynamics operating in *known* environments that are modeled using discrete abstractions, e.g., transition systems. As a result, these methods cannot be applied to scenarios where the environment is initially unknown and, therefore, online re-planning may be required as environmental maps are constructed; resulting in limited applicability. To mitigate this issue, learning-based approaches have also been proposed that consider robots with *unknown* dynamics operating in *unknown* environments. These approaches learn policies that directly map on-board sensor readings to control commands in a trial-and-error fashion; see e.g., [12]–[15]. However, learning approaches, in addition to being data inefficient, are specialized to the environment they were trained on and do not generalize well to previously unseen environments.

This paper addresses a motion planning problem for multiple robots with *known* dynamics that need to accomplish collaborative complex tasks in *unknown* environments. We consider robots that are tasked with accomplishing collaborative tasks captured by global co-safe Linear Temporal Logic (LTL) formulas in complex environments with unknown obstacles. To address this problem, we propose the first reactive and abstraction-free temporal logic planning algorithm for complex mission planning in unknown environments. The robots are equipped with sensors that allow them to continuously learn and update a map of the environment while reacting and adapting mission planning to the updated map. Furthermore, the proposed algorithm is abstraction-free i.e., it does not rely on any discrete abstraction of the robot dynamics (e.g., transition systems). Theoretically, we show that under mild assumptions on the coarse structure of the environment and the sensor model, the proposed algorithm is complete.

Related Research: Temporal logic planning in the presence of map uncertainty in terms of incomplete environment models is also considered in [16]–[23]. For instance, [16], [17] model the environment as transition systems which are partially known. Then controllers are designed by applying graph search methods on a product automaton. As the environment, i.e., the transition system, is updated, the product automaton is locally updated as well, and new paths are re-designed by applying graph search approaches on the revised automaton. A conceptually similar approach is proposed in [20], [21] as well. Methods to locally patch paths as the environment changes so that GR(1) (General Reactivity of Rank 1) specifications, are satisfied are proposed in [18], [19]. Reactive to LTL specifications planning algorithms are proposed in [22], [23], as well. Specifically, in [22], [23] the robot reacts to the environment while the task specification captures this reactivity. Correctness of these algorithms is guaranteed if the robot operates in an environment that satisfies the assumptions that were explicitly modeled in the task specification. Common in all these works is that they rely on discrete abstractions of the robot dynamics – which in general are computationally-expensive to obtain [24], [25] – and on constructing a product automaton among all robots to synthesize motion plans. As a result, these algorithms face scaling challenges as the number of robots grows. Moreover, correctness of the generated motion plans is guaranteed with respect to the discrete abstractions resulting in a gap between the generated discrete motion plans and their physical low-level implementation [26]. To the contrary, our approach does not rely on abstractions of the robot dynamics and, therefore, avoids altogether the construction of a product automaton and scales well with the number of robots.

Contributions: The contribution of this paper can be summarized as follows. *First*, we propose the *first* reactive and abstraction-free temporal logic planning algorithm that can be applied to complex multi-robot mission planning problems in unknown environments. *Second*, we show that the proposed algorithm is complete under assumptions on the coarse structure of the environment and the sensor model. *Third*, we

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.

provide extensive simulation and hardware experiments that corroborate the efficacy of the proposed algorithm.

II. Problem Definition

Consider N mobile robots that reside in an unknown environment $\Omega \subset \mathbb{R}^d$ with non-overlapping but known regions of interest, denoted by $r_i \subset \Omega$, and *static* but *unknown* obstacles $O \subset \Omega$, where d is the dimension of the workspace. The robots are governed by the following dynamics: $\mathbf{p}_j(t+1) = \mathbf{f}_j(\mathbf{p}_j(t), \mathbf{u}_j(t))$, for all $j \in \mathcal{N} := \{1, \ldots, N\}$, where $\mathbf{p}_j(t) \in \mathbb{R}^n$ stands for the state (e.g., position and orientation) of robot j at time $t \geq 0$, $\mathbf{u}_j(t)$. Hereafter, we compactly denote the dynamics of all robots as

$$\mathbf{p}(t+1) = \mathbf{f}(\mathbf{p}(t), \mathbf{u}(t+1)), \tag{1}$$

where $\mathbf{p}(t) \in \mathbb{R}^{n \times N}$, $\forall t \geq 0$.

Moreover, we assume that the robots are equipped with sensors (e.g., LiDAR or camera) to detect if a given location $\mathbf{q} \in \Omega$ belongs to the obstacle-free space. Specifically, we consider the following deterministic sensor model:

$$s(\mathbf{p}_j(t), \mathbf{q}) = \begin{cases} 1 \text{ if } (\mathbf{q} \in O) \wedge (\|\mathbf{p}_j(t) - \mathbf{q}\| \leq R), \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

In words, given a location $\mathbf{q} \in \Omega$ that lies within the sensing range R of robot j, we have that $s(\mathbf{p}_j(t), \mathbf{q}) = 1$ if $\mathbf{q}$ belongs to the obstacle-space, i.e., $\mathbf{q} \in O$; otherwise, $s(\mathbf{p}_j(t), \mathbf{q}) = 0$.

Using sensor measurements as per (2), the robots collectively build a shared (binary) occupancy grid map defined as follows. First, the workspace Ω is partitioned into K grid cells denoted by $c_i \subset \Omega$. Then, given $C = \{c_1, c_2, \ldots, c_K\}$, we define a function a function $g : C \times \mathbf{s}_{0:t} \to \{0,1\}^K$ that maps each grid cell c_i to 0 or 1, given all robot measurements up to time t, denoted by $\mathbf{s}_{0:t}$. The function g is defined as follows:

$$g(c_i, \mathbf{s}_{0:t}) = \begin{cases} 1 \text{ if } \exists j, \mathbf{q} \in c_i,\ t' \leq t \text{ s.t } s(\mathbf{p}_j(t'), \mathbf{q}) = 1, \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

In words, if there exists at least one robot j that took a measurement at any past time instant $t' \leq t$ indicating that there exists an obstacle within the grid cell c_i (i.e., $s(\mathbf{p}_j(t'), \mathbf{q}) = 1$), then $g(c_i, \mathbf{s}_{0:t}) = 1$, i.e., c_i is considered an obstacle. Otherwise, $g(c_i, \mathbf{s}_{0:t}) = 0$, which means that c_i is either obstacle-free or no measurement associated with c_i has been taken up to the current time instant t. As a result, g produces a binary occupancy grid map $M(t) = \{m_1(t), m_2(t), \ldots, m_K(t)\}$, where (i) $m_i(t) = 1$ if $g(c_i, \mathbf{s}_{0:t}) = 1$ (ii) and $m_i(t) = 0$ if $g(c_i, \mathbf{s}_{0:t}) = 0$. Note that the occupancy grid map is constructed as per (3) assuming deterministic and noiseless sensor models as in (2). In practice, to account for noisy sensors, the function g in (3) can be determined by inference methods (e.g., particle filters) giving rise to probabilistic maps which are then thresholded yielding binary maps $M(t)$ as e.g., in [27]–[30]. Moreover, note that for simplicity of notation, we assume that all robots build a shared map $M(t)$. This can be relaxed by allowing each robot to build its own local map $M_j(t)$, based on its individual sensor measurements.

The goal of the robots is to accomplish a complex collaborative task captured by a global Linear Temporal Logic (LTL) specification ϕ. The assigned task ϕ is defined over a set of atomic propositions $\mathcal{AP} = \cup_{\forall j} \mathcal{AP}_j$ where $\mathcal{AP}_j = \cup_i \{\pi_j^{r_i}, \pi_j^O\}$, and $\pi_j^{r_i}$ and π_j^O are atomic predicates that are true when robot j is within region r_i and hits an obstacle, respectively.

In what follows, we briefly present the syntax and semantics of LTL. A detailed overview of this theory can be found in [31]. The basic ingredients of LTL are a set of atomic propositions $\mathcal{AP}$, the boolean operators, i.e., conjunction $\wedge$, and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1\ \mathcal{U}\ \phi_2$, where $\pi \in \mathcal{AP}$. For brevity we abstain from presenting the derivations of other Boolean and temporal operators, e.g., *always* $\Box$, *eventually* $\Diamond$, *implication* $\Rightarrow$, which can be found in [31]. An example of an LTL specification is: $\phi = \Diamond(\pi_1^{r_1} \wedge (\pi_1^{r_2} \wedge (\Diamond \pi_1^{r_3}))) \wedge \Box(\neg(\pi_1^O \vee \pi_1^{r_4}))$, which requires the robot to (i) eventually visit locations r_1, r_2, and r_3 in this order; and (ii) avoid all a priori unknown obstacles and region r_4 since hostile units may exist there.

Hereafter, we assume that the assigned task ϕ is expressed as a *co-safe* LTL formula. Note that co-safe LTL formulas are satisfied by finite-length robot trajectories [31]. Moreover, we exclude the 'next' operator $\bigcirc$ from the syntax, since it is not meaningful for practical robotics applications; this is common in relevant works, see, e.g., [32] and the references therein. In order to interpret a temporal logic formula over the trajectories of the robot system, we use a labeling function $L : \mathbb{R}^{n \times N} \times M(t) \to 2^{\mathcal{AP}}$ that determines which atomic propositions are true given the current robot system state $\mathbf{p}(t) \in \mathbb{R}^{n \times N}$ and the current map $M(t)$ of the environment. Then, the problem addressed in this paper can be summarized as follows:

Problem 1: Given an initial robot configuration $\mathbf{p}(0)$, the robot dynamics (1), sensor models (2), and a task specification captured by a co-safe LTL formula ϕ (which also requires obstacle avoidance), select a horizon H and compute control inputs $\mathbf{u}(t)$ for all time instants $t \in [0, H]$ so that ϕ is satisfied.

III. Temporal Logic Planning in Unknown Environments

Problem 1 can be solved by applying existing abstraction-free approaches for LTL planning, e.g., [9], [33], at runtime to continuously update the robot paths as the map of the environment is continuously learned. Nevertheless, such methods require searching over a large product automaton space, a task that is computationally challenging to perform at runtime. To mitigate this challenge, we propose a new reactive LTL planning algorithm summarized in Algorithm 1 that does not rely on searching over a product automaton space. The main idea of Algorithm 1 is to decompose the temporal logic task into a sequence of local point-to-point planning problems that are formulated and solved online while adapting to the updated map $M(t)$ of the environment.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.

To this end, first, we translate the specification ϕ, constructed using a set of atomic predicates $\mathcal{AP}$, into a Deterministic Finite state Automaton (DFA), defined as follows [31] [line 1, Alg. 1].

Definition 3.1 (DFA): A Deterministic Finite state Automaton (DFA) D over $\Sigma = 2^{\mathcal{AP}}$ is defined as a tuple $D = \left(\mathcal{Q}_D, q_D^0, \delta_D, q_F\right)$, where $\mathcal{Q}_D$ is the set of states, $q_D^0 \in \mathcal{Q}_D$ is the initial state, $\delta_D : \mathcal{Q}_D \times \Sigma \to \mathcal{Q}_D$ is a deterministic transition relation, and $q_F \in \mathcal{Q}_D$ is the accepting/final state.

A run of ρ_D of D over a finite symbol $\sigma = \sigma(1)\sigma(2)\dots\sigma(K) \in (2^{\mathcal{AP}})^*$, is a sequence $\rho_D = q_D^0 q_D^1 q_D^2 \dots, q_D^K$, where $\delta_D(q_D^k, \sigma(k)) = q_D^{k+1}$, $\forall k \in \mathbb{N}$. A run ρ_D is called *accepting* if $q_K = q_F$. A sequence of robot states $\mathbf{p}_{0:H}$ satisfies ϕ, i.e., $\mathbf{p}_{0:H} \models \phi$, if the word $\sigma = L(\mathbf{p}(0), M(0))L(\mathbf{p}(1), M(1)) \dots L(\mathbf{p}(H), M(H))$ yields an accepting DFA run. Hereafter, for simplicity we denote $L(\mathbf{p}(t), M(t))$ by $L(\mathbf{p}(t))$.

In what follows, we denote by $q_D(t) \in \mathcal{Q}_D$ the DFA state that the multi-robot system has reached at time t given all observations that have been made (i.e., all atomic predicates that have been satisfied) until t. We initialize $q_D(0)$ as $q_D(0) = q_D^0$ [line 2, Alg. 1]. The main idea of Algorithm 1 is that at any time t, given the state $q_D(t)$ the robots collaboratively select a DFA state, denoted by q_D^{next}, that they should reach next. The state q_D^{next} is selected so that it is (i) one-hop reachable from $q_D(t)$, (ii) 'closer' to the final state q_F that $q_D(t)$ is according to a distance function defined over the DFA, and (iii) observations that can trigger a transition from $q_D(t)$ to q_D^{next} can be physically generated by the robots (hereafter, such transitions are also called feasible and are defined next). Given the state q_D^{next}, the robots move towards locations in the workspace Ω that will trigger the transition from $q_D(t)$ to q_D^{next}. In the meantime, the robots update the current map $M(t)$ and revise their paths, if needed, to avoid previously unseen obstacles. This process is repeated until the robots reach the the final DFA state q_F, i.e., until the co-safe LTL task is accomplished. In what follows, we describe these steps in detail.

Pruning the DFA: To select q_D^{next} given the current state $q_D(t)$, first we need to define a distance function over the state-space of the DFA that measures how far each state is from the final state. To achieve that, we first prune the DFA D by removing transitions that cannot be enabled [line 3, Alg. 1]. Specifically, the pruned DFA is constructed by removing all infeasible DFA transitions. To formally define the *feasible/infeasible* DFA transitions we first need to introduce the following definitions.

Definition 3.2 (Feasible symbols $\sigma_j \in \Sigma_j$): A symbol $\sigma_j \in \Sigma_j := 2^{\mathcal{AP}_j}$ is *feasible* if and only if $\sigma_j \not\models b_j^{\text{inf}}$, where b_j^{inf} is a Boolean formula defined as

$$b_j^{\text{inf}} = \vee_{\forall r_i}(\vee_{\forall r_e}(\pi_j^{r_i} \wedge \pi_j^{r_e})). \tag{4}$$

where b_j^{inf} requires robot j to be present simultaneously in more than one disjoint region.

Note that the Boolean formula b_j^{inf} is satisfied by any finite symbol $\sigma_j \in \Sigma_j$ that requires robot j to be present in two or more disjoint regions, simultaneously. For instance, the symbol $\sigma_j = \pi_j^{r_i}\pi_j^{r_e}$ satisfies b_j^{inf} if r_i and r_e are disjoint regions. Next, we define the feasible symbols $\sigma \in \Sigma$.

Definition 3.3 (Feasible symbols $\sigma \in \Sigma$): A symbol $\sigma \in \Sigma$ is *feasible* if and only if $\sigma_j \not\models b_j^{\text{inf}}$, for all robots j, where $\sigma_j = \Pi|_{\Sigma_j}\sigma$, b_j^{inf} is defined in (4), and $\Pi|_{\Sigma_j}\sigma$ stands for the projection of the symbol σ onto Σ_j.[1] We denote by $\Sigma_{\text{feas}} \subseteq \Sigma$ the set of all feasible symbols $\sigma \in \Sigma$.

Moreover, let $b_{\text{inv}}^{q_D}$ denote the Boolean formula that needs to be satisfied to enable a loop around q_D, i.e., a transition from q_D to q_D.[2] Let also $\Sigma_{\text{inv}}^{q_D} \subseteq \Sigma$ be a set that collects all symbols that satisfy $b_{\text{inv}}^{q_D}$. Let also $b_{\text{next}}^{q_D,q_D'}$ denote the Boolean formula that needs to be satisfied to enable a transition from q_D to q_D' and $\Sigma_{\text{next}}^{q_D,q_D'} \subseteq \Sigma$ be a set that collects all symbols that satisfy $b_{\text{next}}^{q_D,q_D'}$. Also, from $\Sigma_{\text{text}}^{q_D,q_D'}$, we exclude symbols associated with robots that do not play any role in satisfying $b_{\text{next}}^{q_D,q_D'}$. For instance, if $b_{\text{next}}^{q_D,q_D'} = \pi_j^{r_e} \vee \pi_j^{r_m}$, then $\Sigma_{\text{text}}^{q_D,q_D'}$ is defined as $\Sigma_{\text{text}}^{q_D,q_D'} = \{\pi_j^{r_e}, \pi_j^{r_m}, \pi_j^{r_e}\pi_j^{r_m}\}$, i.e., it does not include any symbol associated with other robots $i \neq j$ that may also satisfy $b_{\text{next}}^{q_D,q_D'}$ (as e.g., $\pi_j^{r_e}\pi_i^{r_f}$). The same also holds for $\Sigma_{\text{inv}}^{q_D}$; see also Example 3.5. Next, given any pair of symbols $\sigma_{\text{inv}} \in \Sigma_{\text{inv}}^{q_D}$ and $\sigma_{\text{next}} \in \Sigma_{\text{next}}^{q_D,q_D'}$, we define the sets $\mathcal{R}_{\text{inv}} \subseteq \mathcal{N}$ and $\mathcal{R}_{\text{next}} \subseteq \mathcal{N}$ that collect the indices j of the robots that are involved in generating σ_{inv} and σ_{next}, respectively. Also, let L_j^{inv} denote the location that robot $j \in \mathcal{R}_{\text{inv}}$ should be located to generate σ_{inv}. Similarly, we define L_j^{next}. For instance if $\sigma_{\text{next}} = \pi_1^{r_2}\pi_3^{r_2}$, then we have that $\mathcal{R}_{\text{next}} = \{1, 3\}$, $L_1^{\text{next}} = r_2$, and $L_3^{\text{next}} = r_2$.

Given the above definitions, we define the feasible DFA transitions as follows; see also Example 3.5.

Algorithm 1: LTL Planning in Unknown Environments

Input: (i) initial robot configuration $\mathbf{p}(0)$, (ii) LTL formula ϕ

Convert ϕ into a DFA;
Initialize $t = 0$, $q_D(0) = q_D^0$;
Prune DFA;
Select the next DFA state q_D^{next} that should be reached;
Compute L_j^{next} and the set of robots $\mathcal{R}_{\text{next}}$;
while $q_D(t) \neq q_F$ **do**
  for $j \in \mathcal{R}_{next}$ **do**
    Compute $\mathbf{p}_{j,t:T_j}$ as per (6);
  Execute paths $\mathbf{p}_{j,t:T_j}$, re-plan if needed, and update map $M(t)$;
  if $t = T$ **then**
    Update $q_D(t) = q_D^{\text{next}}$;
    Select a new DFA state q_D^{next};
    Compute new L_j^{next} and the set of robots $\mathcal{R}_{\text{next}}$;

[1] For instance, $\Pi|_{\Sigma_j}(\pi_j^{r_e}\pi_m^{r_h}) = \pi_j^{r_e}$.

[2] Note that $b_{\text{inv}}^{q_D}$ exists for all states q_D, since the 'next' operator is excluded from the syntax. Also, the Boolean formulas that should be satisfied to enable a transition from a state q_D to q_D' can be provided by tools that convert temporal logic formulas to automata.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.

Definition 3.4 (Feasible DFA transitions): A transition $\delta_D(q_D,\cdot)=q'_D$ is feasible if (i) $\Sigma^{q_D}_{\text{inv}}\cap\Sigma_{\text{feas}}\neq\emptyset$, and (ii) for all symbols $\sigma^{q_D}_{\text{inv}}\in\Sigma^{q_D}_{\text{inv}}\cap\Sigma_{\text{feas}}\neq\emptyset$ there exists at least one symbol $\sigma^{q_D,q'_D}_{\text{next}}\in\Sigma^{q_D,q'_D}_{\text{next}}\cap\Sigma^{q'_D}_{\text{inv}}\cap\Sigma_{\text{feas}}$ such that either (a) $\mathcal{R}^{q_D}_{\text{inv}}\cap\mathcal{R}^{q_D,q'_D}_{\text{next}}=\emptyset$ or (b) $\mathcal{R}^{q_D}_{\text{inv}}\cap\mathcal{R}^{q_D,q'_D}_{\text{next}}\neq\emptyset$ and $L^{\text{next}}_j=L^{\text{inv}}_j$.

Then, we prune the DFA by removing transitions that are not feasible, as per Definition 3.4. In words, a transition from q_D to q'_D is feasible and, therefore, it is not removed (a) if the set $\Sigma^{q_D}_{\text{inv}}\cap\Sigma_{\text{feas}}$ is non-empty, i.e., there are feasible symbols that allow the robots to stay in state q_D; and (ii) for each possible symbol in $\Sigma^{q_D}_{\text{inv}}\cap\Sigma_{\text{feas}}$ that the robots may generate to reach q_D, there exists a feasible symbol that will enable the robots to eventually reach a new state q'_D from q_D and stay in q'_D.[3] Also, in the latter condition, we require the set of robots (denoted by $\mathcal{R}^{q_D,q'_D}_{\text{next}}$) that are involved in satisfying the Boolean condition $b^{q_D,q'_D}_{\text{next}}$ to be either (a) disjoint from the set of robots (denoted by $\mathcal{R}^{q_D}_{\text{inv}}$) required to to satisfy $b^{q_D}_{\text{inv}}$ or (b) if there exist robots $j\in\mathcal{R}^{q_D}_{\text{inv}}\cap\mathcal{R}^{q_D,q'_D}_{\text{next}}\neq\emptyset$ then these robots should suffice to stay in their current locations, i.e., $L^{\text{next}}_j=L^{\text{inv}}_j$, to 'contribute' to satisfaction of $b^{q_D,q'_D}_{\text{next}}$. This ensures that transition from q_D to q'_D can be cast as *local* reachability problems, where each robot involved in $\mathcal{R}^{q_D,q'_D}_{\text{next}}$ needs to move to a region of interest while avoiding certain regions and obstacles.

On the pruned DFA, we define a function $d:\mathcal{Q}_D\times\mathcal{Q}_D\to\mathbb{N}$ that returns the minimum number of *feasible* DFA transitions that are required to reach a state $q'_D\in\mathcal{Q}_D$ starting from a state $q_D\in\mathcal{Q}_D$. Particularly, we define the distance function $d:\mathcal{Q}_D\times\mathcal{Q}_D\to\mathbb{N}$ as follows

$$d(q_D,q'_D)=\begin{cases}|SP_{q_D,q'_D}|, & \text{if } SP_{q_D,q'_D} \text{ exists,}\\ \infty, & \text{otherwise,}\end{cases}\tag{5}$$

where SP_{q_D,q'_D} denotes the shortest path (in terms of hops) in the pruned DFA from q_D to q'_D and $|SP_{q_D,q'_D}|$ stands for its cost (number of hops).

Example 3.5 (Feasible DFA transitions): (a) Consider the case, where $b^{q_D}_{\text{inv}}=1$ and $b^{q_D,q'_D}_{\text{next}}=\pi^{r_m}_j$. By definition of $\Sigma^{q_D}_{\text{inv}}$ and $\Sigma^{q_D,q'_D}_{\text{next}}$, we have that $\Sigma^{q_D}_{\text{inv}}=\{\varnothing\}$, where $\varnothing$ stands for the empty symbol, and $\Sigma^{q_D,q'_D}_{\text{next}}=\{\pi^{r_m}_j\}$. Transition from q_D to q'_D is feasible as per Definition 3.4. Recall that the sets $\Sigma^{q_D}_{\text{inv}},\Sigma^{q_D,q'_D}_{\text{next}}$ collect symbols that are associated with robots that play a role in satisfying the respective Boolean formulas. This precludes cases where, e.g., a symbol $\pi^{r_e}_j$, $e\neq m$ is included in $\Sigma^{q_D}_{\text{inv}}$ resulting in erroneously characterizing the transition from q_D to q'_D as infeasible due to condition (ii). (b) Consider also the Boolean formulas $b^{q_D}_{\text{inv}}=b^{q'_D}_{\text{inv}}=\pi^{r_i}_j$ and $b^{q_D,q'_D}_{\text{next}}=\pi^{r_e}_j\wedge\pi^{r_h}_m$, then the transition from q_D to q'_D is infeasible, since it holds that $j\in\mathcal{R}^{q_D}_{\text{inv}}\cap\mathcal{R}^{q_D,q'_D}_{\text{next}}$ and, $L^{\text{next}}_j\neq L^{\text{inv}}_j$. In fact, transition from q_D to q'_D would require robot j to instantaneously move from r_i to r_e which is not possible given that r_i and r_e are disjoint regions.

[3]Note that Definition 3.4 is conservative due to condition (ii). In fact, there may exist feasible symbols $\sigma^{q_D}_{\text{inv}}$ and $\sigma^{q_D,q'_D}_{\text{next}}$ to enable a transition from q_D to q'_D even if (ii) is not satisfied. Conservative pruning ensures that when the robots reach a state q_D, they will always be able to make progress towards accomplishing their task (i.e., reach a new state q'_D) regardless of the symbols they have generated to reach q_D; see also Section IV.

Selecting the Next DFA State: The pruning process described before occurs offline, i.e., before deploying the robots in the workspace. In what follows, we present the online part of Algorithm 1. Given the current robot positions $\mathbf{p}_j(t)$ and the current DFA state $\mathbf{q}_D(t)$ at time $t\geq 0$, the robots coordinate to select a new DFA state, denoted by q^{next}_D, that they should reach next. The DFA state q^{next}_D is selected as follows. First, we compute the distance $d(q_D(t),q_F)$. Then, we select as q^{next}_D a state that satisfies $d(q^{\text{next}}_D,q_F)=d(q_D(t),q_F)-1$, i.e., a state that is one hop closer to q_F than $q_D(t)$ is. If there is more than one state that satisfies this condition, we select one randomly [line 4, Alg. 1]. The state q^{next}_D is updated when the robots reach it, i.e., at a time instant $t'\geq t$ that satisfies $q_D(t')=q^{\text{next}}_D$ [lines 10-12, Alg. 1].

Local Construction of Robot Paths: Given the DFA state q^{next}_D, our goal is to design paths $\mathbf{p}_{j,t:T_j}$, for all robots j that should be followed within a time interval $[t,T_j]$ so that the DFA transition from $q_D(t)$ to q^{next}_D is activated. To achieve that, it suffices to ensure that the robots follow paths that (i) satisfy the Boolean formula $b^{q_D(t)}_{\text{inv}}$ that allows them to stay in the current DFA state $q_D(t)$, until (ii) the Boolean formula $b^{q_D(t),q^{\text{next}}_D}_{\text{next}}$ that enables transition from $q_D(t)$ to q^{next}_D is satisfied while (iii) never satisfying properties that enable transitions to DFA states $q'_D\neq q_D(t)$ and $q'_D\neq q^{\text{next}}_D$. For simplicity, in what follows, we replace $b^{q_D(t)}_{\text{inv}}$ and $b^{q_D(t),q^{\text{next}}_D}_{\text{next}}$ with b_{inv} and b_{next} respectively. The same notation extends to $\sigma^{q_D(t)}_{\text{inv}}$ and $\sigma^{q_D(t),q'_D}_{\text{next}}$ as well.

To design paths $\mathbf{p}_{j,t:T_j}$, first the robots *communicate* to select a symbol σ_{next} that satisfies condition (ii) of Definition 3.4; see also Example 3.6. Note that such a symbol exists by construction of the pruned DFA for any sequence of atomic predicates that was generated to reach $q_D(t)$. Among all such symbols σ_{next}, we select the one that yields the smallest set $\mathcal{R}_{\text{next}}$ to minimize the number of robots required to navigate the workspace to reach the state q^{next}_D. Given σ_{next}, we compute the regions L^{next}_j that the robots $j\in\mathcal{R}_{\text{next}}$ should visit to generate the symbol σ_{next}. We design the paths $\mathbf{p}_{j,t:T_j}$ towards L^{next}_j as follows. First, all robots $j\notin\mathcal{R}_{\text{next}}$ stay put at their current locations until the state q^{next}_D is reached. Every other robot $j\in\mathcal{R}_{\text{next}}$ locally solves the following feasibility problem to design its path $\mathbf{p}_{j,t:T_j}$ [lines 7-8, Alg. 1].

$$\text{Compute: } T_j,\mathbf{u}_{j,t:T_j} \text{ such that}\tag{6a}$$
$$\mathbf{p}_j(t')\notin\cup_i\{r_i\}\setminus(R^t_j\cup L^{\text{next}}_j),\forall t'\in[t,T_j]\tag{6b}$$
$$\mathbf{p}_j(t')\notin\hat{O}(t),\forall t'\in[t,T_j]\tag{6c}$$
$$\mathbf{p}_j(T_j)\in L^{\text{next}}_j\tag{6d}$$
$$\mathbf{p}_j(t'+1)=\mathbf{f}_j(\mathbf{p}_j(t'+1),\mathbf{u}_j(t'+1)),\tag{6e}$$

In (6b), R^t_j denotes the region that robot j lies in at time t. The first constraint (6b) requires that robot j should not visit any region r_i excluding R^t_j and the target region L^{next}_j, for all

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.

$t' \in [t, T_j]$. If at time t robot j does not lie in any of the regions r_i, then $R_j^t = \emptyset$. Similarly, the second constraint in (6c) requires robot j to avoid all obstacles, collected in $\hat{O}(t)$, that are known up to the time instant t. The third constraint in (6d) requires robot j to be in region L_j^{next} at a time instant $T_j \geq t$. This is required to generate the symbol σ_{next}. The last constraint in (6e) captures the robot dynamics. Note that among all paths that satisfy these constraints, the optimal one, as a per a user-specified motion cost function, can be selected. Observe that (6) is a point-to-point navigation problem, and, therefore, any of the traditional motion planning algorithms, such as A*, RRT, RRT*, RRG, or PRM, can be used to solve it [34]. Finally observe that given L_j^{next}, robot j locally solves (6) without communicating with any other robot.

Example 3.6 (Selection of σ_{next} and σ_{inv}): Consider the Boolean formulas $b_{\text{inv}} = \neg\pi_{\text{obs}}$ and $b_{\text{next}} = (\pi_j^{r_1} \vee \pi_j^{r_3}) \wedge \pi_e^{r_2} \wedge (\neg\pi_j^{r_2}) \wedge (\neg(\pi_j^{r_1} \wedge \pi_e^{r_1}))$, where the atomic predicate π_{obs} is true of at least one robot hits an obstacle. A symbol σ_{next} that satisfies b_{next} is $\sigma_{\text{next}} = \pi_j^{r_1}\pi_e^{r_2}$ while b_{inv} is satisfied by an empty string, i.e., $\sigma_{\text{inv}} = \varnothing$. Given this pair of symbols, we have that $\mathcal{R}_{\text{inv}} = \emptyset$, $\mathcal{R}_{\text{next}} = \{j, e\}$, $L_j^{\text{next}} = r_1$, and $L_e^{\text{next}} = r_2$.

Reactive Planning to Map Uncertainty: The robots need to adapt planning to the continuously learned occupancy grid map. In particular, first, as the robots follow their paths $\mathbf{p}_{j,t:T_j}$, they may discover new obstacles based on their sensor measurements; see (2). If the discovered obstacles intersect with $\mathbf{p}_{j,t:T_j}$, for some $j \in \mathcal{R}_{\text{next}}$, then that robot j locally re-solves (6) using the updated map to design a new path. Note that existing planning algorithms that can address reachability navigation problems in the presence of unknown obstacles can also be used to solve (6); see e.g., [35]–[38].

Second, as the robots update the map $M(t)$ while following their designed paths, they may realize that their goal regions L_j^{next} are not reachable because of the structure of the environment (e.g., regions of interest may be surrounded by obstacles). In this case, the robot j that cannot reach L_j^{next} communicates with all other robots so that a new symbol σ_{next} is selected. This process is repeated until a reachable region L_j^{next} is selected. If such a symbol σ_{next} does not exist, then this means that the transition from $q_D(t)$ to q_D^{next} is infeasible, given the current multi-robot state $\mathbf{p}(t)$, due to the a-priori unknown structure of the environment. Then, the pruned DFA is updated by removing this transition. Next, a new target DFA state is selected that is closer to the final state q_F than $q_D(t)$ is. If there do not exist such DFA states then the robots cannot accomplish the assigned task. Note that this does not necessarily mean that Problem 1 is infeasible. In fact, there may exist another sequence of DFA states for which Problem 1 is always feasible. Conditions under which (6) (and, consequently, Problem 1) is always feasible (i.e., paths towards selected goal regions always exist) and, therefore, Algorithm 1 is complete, are provided in Section IV.

Asynchronous Execution of Paths: The robots $j \in \mathcal{R}_{\text{next}}$ execute their paths $\mathbf{p}_{j,t:T_j}$ *asynchronously* until the time instant $T_j - 1$, i.e., right before the robots step into their respective target regions L_j^{next} [line 9, Alg. 1]. Then, every robot $j \in \mathcal{R}_{\text{next}}$ waits for all other robots $e \in \mathcal{R}_{\text{next}}$ with $T_e > T_j$. At the time instant $T-1$, where $T = \max_j\{T_j\}$, they *synchronously* move towards L_j^{next} and the symbol σ_{next} is generated. As it will be shown in Section IV, the robots remain in the DFA state $q_D(t)$ for all $t' \in [t, T-1]$. At the time instant T the symbol σ_{next} is generated and transition from $q_D(t)$ to q_D^{next} occurs. Once this happens, the robots update their current DFA state and select the next DFA state [lines 10-13]. This process is repeated until the final state q_F is reached.

Remark 3.7 (Communication): Execution of the paths $\mathbf{p}_{j,t:T_j}$ does not require all-time communication among robots, which is typically the case in relevant temporal logic algorithms. Instead, communication is intermittent and is required only in the following two cases: (i) all robots need to communicate with each other to select the symbol σ_{next} (ii) every robot $j \in \mathcal{R}_{\text{next}}$ needs to send a message to all other robots in $\mathcal{R}_{\text{next}}$ once it is ready to step into the region L_j^{next}.

IV. Algorithm Analysis

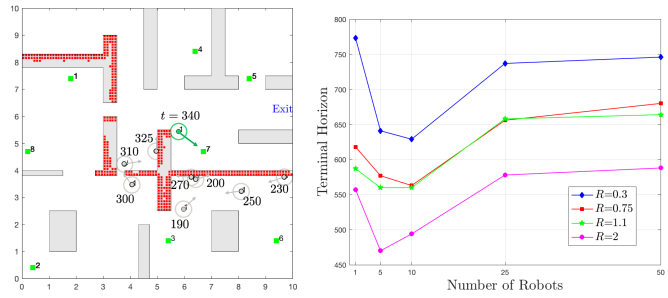
To prove that Algorithm 1 is complete, we need first to state the following Lemma. The proof is omitted due to space limitations.

Lemma 4.1: Let q_D be the DFA state at time t and $\mathbf{p}_{j,t:T_j}$ be the paths that the robots designed to reach q_D^{next}. If all robots execute the paths $\mathbf{p}_{j,t:T_j}$, as per Section III, and if all robots $j \in \mathcal{R}_{\text{next}}$ reach their assigned regions L_j^{next}, then the robots remain in q_D during the time interval $[t, T-1]$ and reach q_D^{next} at the time instant $T = \max_j\{T_j\}$.

Proposition 4.2 (Completeness): Assume that there exists a path from the initial to the final state of the pruned DFA that corresponds to the assigned co-safe temporal logic task. Assume also that there exists at least one path that connects any location $\mathbf{q} \in \Omega$ to any region r_i without crossing any region $r_e \neq r_i$ and obstacles. If at least one of these paths respects the robot dynamics (1), and (2) models an omnidirectional sensor, then Algorithm 1 is complete, i.e., if there exists a solution to Problem 1, then Algorithm 1 will find it.

Proof: To show this result it suffices to show that eventually the accepting condition of the DFA is satisfied, i.e., the robots will reach the final states q_F. First, recall that the robots select only feasible DFA states q_D^{next}. Thus to ensure that q_F will be reached it suffices to show that (i) the optimal control problem (6) is always feasible. and (ii) by following the designed paths $\mathbf{p}_{j,t:T_j}$, designed as per (6), the state q_D^{next} will be reached. To prove (i), note that (6) is always feasible, since, by assumption, there always exists a path that connects the current robot location $\mathbf{p}_j(t)$ to the region L_j^{next} (determined by σ_{next}) while avoiding obstacles and other regions and respecting the robot dynamics, as captured by the constraints of (6). To prove (ii), first notice that since by assumption the robot is equipped with an omnidirectional sensor as per (2), it will always be aware of all obstacles within range R. In other words, if the path $\mathbf{p}_{j,t:T_j}$ passes through an obstacle, robot j will always detect it and will re-solve (6) to update its path.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.



(a) $t = 340$ (b) Effect of Sensing Radius

Fig. 1. Case $N = 1$, $R = 0.3$: Figure 1(a) illustrate the robot position (discs) and actions (arrows) while it is heading towards region 7. The green circle denotes the sensing range. At time $t = 190$ the robot is not aware of the horizontal obstacle that is between it and region 7. As a result it moves toward the obstacle, until it is fully discovered. Then, a new obstacle-free path is eventually discovered to reach region 7 ($t = 340$). The red squares stands for the grid cells of the map $M(t)$ that are occupied by obstacles.

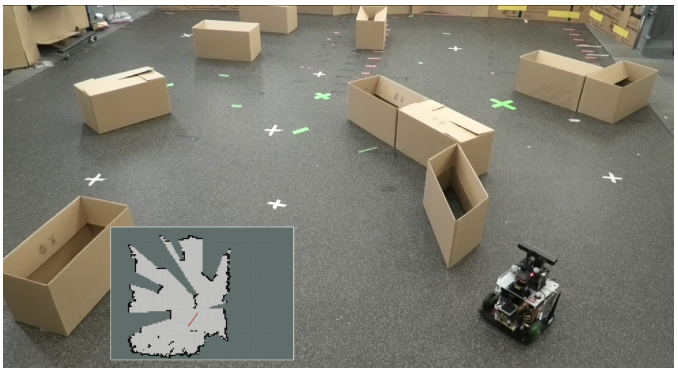


Fig. 2. Experiment setup: A differential drive robot equipped with a LiDAR sensor navigates an unknown indoor environment to accomplish a complex mission.

Since, the obstacles are static and (6) is always feasible as discussed before, the region L_j^{next} will eventually be reached. Also, due to Lemma 4.1, we have that the robots remain in the DFA state $q_D(t)$, during the time interval $[t, T-1]$ and at the time instant $T = \max_j\{T_j\}$, i.e., when all robots $j \in \mathcal{R}_{\text{next}}$ reach their respective L_j^{next}, transition from $q_D(t)$ to q_D^{next} is enabled completing the proof. ■

V. Numerical Simulations & Experiments

In this section, we present simulations and experiments that illustrate the efficiency of the proposed algorithm.

Simulations: In our simulations, we consider a team of $N \in \{1, 5, 10, 25, 50\}$ robots with differential drive dynamics that reside in an unknown 10×10 environment; see Fig. 1. The robots are equipped with omnidirectional sensors of limited range $R \in \{0.3, 0.75, 1.1, 2\}$. The robots are tasked with providing delivery services to known locations, in a user-specified order, while always avoiding obstacles and dangerous regions. Once and only when services are provided, the robots should exit the building. This task is captured by the following LTL formula $\phi = \Diamond(\xi_1 \wedge \Diamond(\xi_2 \wedge \Diamond(\xi_3 \wedge (\Diamond\xi_4 \wedge (\Diamond\xi_5))))) \wedge (\wedge_{e\in\{1,2,3,4,5\}}(\neg\xi_{\text{exit}}\mathcal{U}\xi_e)) \wedge (\Box\neg\xi_6) \wedge (\Box\neg\xi_{\text{obs}})$ where ξ_e is a Boolean formula that requires a sub-team of robots to visit a specific region to provide e.g., supplies, for all $e \in \{1, \ldots, 6\}$, while ξ_{exit} is true if all robots visit the exit of the building. For instance, when $N = 1$, the Boolean formulas ξ_e are defined as: (i) $\xi_1 = (\pi_1^8 \vee \pi_1^5)$; (ii) $\xi_2 = \pi_1^7$; (iii) $\xi_3 = \pi_1^4$; (iv) $\xi_4 = (\pi_1^1 \vee \pi_1^2)$; (v) $\xi_5 = \pi_1^6$; and (vi) $\Box\neg\xi_6$ requires the robot to always keep a distance of at least 0.3 from region 3. The runtime required to prune the corresponding DFA was less than a second. Observe in Figure 1(a) that as the sensing range R increases, the terminal horizon H that solves Problem (1) decreases, since the larger the sensing range, the faster the robots can learn the map of the environment. The average time required by each robot to solve (6) using RRT was 0.12 seconds approximately. The simulation videos can be found in [39]. Note that an alternative approach to solve the considered planning problem in unknown environments is to use existing abstraction-free approaches for LTL planning [9], [33] to continuously update the robot paths as the map of the environment is learned. Nevertheless, such methods require searching over a large product automaton space to design paths that reach the final DFA state, a task that is computationally challenging to perform at runtime. To the contrary, the proposed method is more efficient as it searches over only the physical state-space to design paths that solve reachability planning problems; see (6).

Experiments: The proposed sensor-based reactive planning algorithm was also evaluated experimentally. Specifically, Alg. 1 was implemented in MatLab with ROS and run on the Scarab platform developed at the University of Pennsylvania [40]. The Scarab is a differential drive robot equipped with a Hokuyo UTM-30LX scanning laser that can obtain range measurements in a 270^0 field-of-view up to a distance of 30m. Laser-based sensing is used for robot localization and for building an occupancy grid map using the gmapping ROS package implementing the SLAM approach presented in [30].

We validated Alg. 1 in complex indoor environments with *unknown* and *static* obstacles; see e.g., Fig. 2. The task of the robot is captured by the following LTL formula: $\phi = \Diamond(\pi_1^{r_4}) \wedge \Diamond(\pi_1^{r_1}) \wedge \Diamond(\pi_1^{r_2} \vee \pi_1^{r_5}) \wedge \Diamond(\pi_1^{r_6} \vee \pi_1^{r_4}) \wedge \Diamond(\pi_1^{r_7} \vee \pi_1^{r_5}) \wedge (\neg\pi_1^{r_1}\mathcal{U}\pi_1^{r_3}) \wedge (\Box\neg\xi^{\text{obs}})$, which requires the robot to eventually visit certain regions of interest while always avoiding obstacles. Note that in this experiment setup, we do not consider an omnidirectional sensor as required in Proposition 4.2; nevertheless, the robot successfully accomplished the assigned mission in two different environments. A video with the conducted experiments can be found in [41].

VI. Conclusion

In this paper, we proposed a new reactive and abstraction-free temporal logic planning approach for multi-robot systems that reside in unknown environments. The robots are tasked with accomplishing complex tasks, captured by temporal logic formulas. The robots are equipped with sensors that allow them to continuously learn and update a map of the unknown environment while adapting mission planning to it. Theoretical guarantees along with mumerical simulations and hardware experiments support the proposed framework.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3116–3121.

[2] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.

[3] J. Tumova and D. V. Dimarogonas, "Multi-agent planning under local ltl specifications and event-based synchronization," *Automatica*, vol. 70, pp. 239–248, 2016.

[4] 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, 2012.

[5] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[6] A. Ulusoy, S. L. Smith, and C. Belta, "Optimal multi-robot path planning with ltl constraints: guaranteeing correctness through synchronization," in *Distributed Autonomous Robotic Systems*. Springer, 2014, pp. 337–351.

[7] Y. Shoukry, P. Nuzzo, A. Balkan, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Linear temporal logic motion planning for teams of underactuated robots using satisfiability modulo convex programming," in *IEEE 56th Annual Conference on Decision and Control (CDC)*, December 2017, pp. 1132–1137.

[8] Y. Shoukry, P. Nuzzo, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, and P. Tabuada, "Smc: Satisfiability modulo convex programming [40pt]," *Proceedings of the IEEE*, no. 99, pp. 1–25, 2019.

[9] C. I. Vasile and C. Belta, "Sampling-based temporal logic path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, November 2013, pp. 4817–4822.

[10] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multi-robot systems under global temporal tasks," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2018.

[11] ——, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *arXiv preprint arXiv:1809.08345*, 2018.

[12] 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 *CDC*, 2014, pp. 1091–1096.

[13] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *IROS*. IEEE, 2017, pp. 3834–3839.

[14] A. Jones, D. Aksaray, Z. Kong, M. Schwager, and C. Belta, "Robust satisfaction of temporal logic specifications via reinforcement learning," *arXiv preprint arXiv:1510.06460*, 2015.

[15] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 2019, pp. 237–248.

[16] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013, pp. 5025–5032.

[17] 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.

[18] S. C. Livingston, R. M. Murray, and J. W. Burdick, "Backtracking temporal logic synthesis for uncertain environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 5163–5170.

[19] S. C. Livingston, P. Prabhakar, A. B. Jose, and R. M. Murray, "Patching task-level robot controllers based on a local μ-calculus formula," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4588–4595.

[20] M. R. Maly, M. Lahijanian, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal motion planning for hybrid systems in partially unknown environments," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 2013, pp. 353–362.

[21] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 583–599, 2016.

[22] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[23] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[24] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, 2005.

[25] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[26] A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari, "Soter: A runtime assurance framework for programming safe robotics systems."

[27] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[28] S. Thrun, "Learning occupancy grids with forward models," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 3. IEEE, 2001, pp. 1676–1681.

[29] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[30] G. Grisetti, C. Stachniss, W. Burgard *et al.*, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, p. 34, 2007.

[31] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press Cambridge, 2008, vol. 26202649.

[32] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[33] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multi-robot temporal logic optimal control synthesis," *arXiv preprint arXiv:1909.00526*, 2019.

[34] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[35] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. Di Caro, "Human-friendly robot navigation in dynamic environments," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6-10, 2013, pp. 423–430.

[36] V. Vasilopoulos, W. Vega-Brown, O. Arslan, N. Roy, and D. E. Koditschek, "Sensor-based reactive symbolic planning in partially known environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 1–5.

[37] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," *arXiv preprint arXiv:1903.02531*, 2019.

[38] M. Ryll, J. Ware, J. Carter, and N. Roy, "Efficient trajectory planning for high speed flight in unknown environments," in *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada, May 2019, pp. 732–738.

[39] Simulation_Video, https://vimeo.com/358545200.

[40] N. Michael, J. Fick, and V. Kumar, "Experimental testbed for large multirobot teams," 2008.

[41] Experiment_Video, https://vimeo.com/394649017.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:16:44 UTC from IEEE Xplore. Restrictions apply.