Hierarchical Temporal Logic Specifications for Abstract Safety Tasks

Xusheng Luo[1] and Changliu Liu[1]

***Abstract*—Robots operating in human environments must satisfy abstract safety constraints—subtle behaviors that go beyond avoiding collisions, such as not reaching over laptops when handling liquids. These constraints are difficult to formalize with traditional task planning methods. We propose Hierarchical LTL on finite traces (H-LTL$_f$) as a specification language for such tasks, enabling modular and interpretable task definitions. To support this, we develop a bottom-up planning algorithm that performs Simultaneous Task Allocation and Planning (STAP) over structured task graphs. Our approach improves scalability and is validated through scenarios involving service robots under complex safety requirements.**

I. Introduction

Imagine a typical office scenario: a human working at a desk asks a humanoid robot to refill their water. The robot approaches the desk to retrieve the cup, and two possible behaviors emerge. In the first, the robot reaches its arm directly over the laptop to grab the cup. In the second, it walks to the side of the desk before extending its arm. Clearly, the latter is more desirable—it minimizes the risk of damaging the laptop in case the cup slips or water spills. This situation raises an interesting question: how can we formally and abstractly specify these types of safety constraints, which are more subtle than conventional collision avoidance?

Formal methods, known for their mathematical rigor, offer powerful tools to specify, develop, and verify both hardware and software systems [1]. One promising direction is to employ formal specification languages—such as Linear Temporal Logic (LTL)—to express high-level task requirements. LTL is based on atomic propositions that can be grounded in the robot's environment, and it supports rich temporal and logical expressions capable of capturing complex task-level constraints. In this work, we explore service robotics tasks that are subject to various safety-related constraints. For instance, a robot collecting trash should avoid crowded public areas, and a robot recording a meeting should ensure the camera is turned off when outside the meeting room. While temporal logic offers expressive power for task specification, it also introduces significant computational challenges. For example, encoding an LTL formula typically requires translating it into an automaton—a graphical representation of the logic. As shown in [2], a task involving the collection of five keys followed by opening five doors resulted in an automaton with 65 states and 792 edges, and the conversion alone took nearly 30 minutes. Similarly, in our experiments, we found that generating an automaton for a complex task could not be completed within an hour. This computational bottleneck stems from the standard approach of encoding all task requirements into a single, flat LTL formula. Such monolithic specifications become intractable as task complexity grows. However, many robot tasks exhibit a natural modularity and can be decomposed into loosely coupled subtasks. Cognitive studies indicate that humans also prefer hierarchical task representations, as they enhance interpretability, allow for easier tracking of progress, and simplify local modifications without disrupting the entire plan [3, 4].

Motivated by this, we introduce a hierarchical extension of a widely used formalism—Linear Temporal Logic on finite traces (LTL$_f$) [5]. Unlike standard LTL, LTL$_f$ is satisfied over finite sequences, making it particularly well-suited for modeling and reasoning about finite-horizon tasks commonly encountered in robotics. In this paper, we first formalize the syntax and semantics of Hierarchical LTL$_f$ (H-LTL$_f$). We then develop a bottom-up planning algorithm that operates over a product graph combining the environment and the task structure. This approach enables Simultaneous Task Allocation and Planning (STAP) and improves scalability by leveraging the inherent structure of robot tasks.

Contributions The contributions are listed as follows:

1) We introduce a hierarchical form of LTL$_f$ that is capable of specifying abstract safety tasks;
2) We develop a search-based planning algorithm, achieving simultaneous task allocation and planning for multi-robot systems;
3) We conduct extensive comparative simulations focusing on service tasks to showcase the efficacy of the hierarchical temporal logic specifications.

II. Preliminaries

Notation: Let $\mathbb{N}$ denote the set of all integers, $[K] = \{0, \dots, K\}$ and $[K]_+ = \{1, \dots, K\}$ represent the sets of integers from 0 to K and from 1 to K, respectively, and $|\cdot|$ denote the cardinality of a set.

LTL [6] is a type of formal logic whose basic ingredients are a set of atomic propositions $\mathcal{AP}$, the Boolean operators, conjunction $\wedge$ and negation $\neg$, and temporal operators, next $\bigcirc$ and until $\mathcal{U}$. LTL formulas over $\mathcal{AP}$ abide by the grammar

$$\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc \phi \mid \phi_1 \,\mathcal{U}\, \phi_2. \quad (1)$$

For brevity, we abstain from deriving other Boolean and temporal operators, e.g., *disjunction* $\vee$, *implication* $\Rightarrow$, *always* $\Box$, *eventually* $\Diamond$, which can be found in [6].

LTL$_f$ [5] is a variant of LTL that is interpreted over finite sequences of states while maintaining the same syntax as standard LTL. A finite *word* σ over the alphabet $2^{\mathcal{AP}}$ is defined as a finite sequence $\sigma = \sigma_0\sigma_1 \dots \sigma_h$ with $\sigma_i \in 2^{\mathcal{AP}}$ for $i \in [h]$. The satisfaction of an LTL$_f$ formula ϕ over a sequence σ at an instant i, for $i \in [h]$, is inductively defined as follows:

[1]Xusheng Luo and Changliu Liu are with Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA (e-mail: {xushengl, cliu6}@andrew.cmu.edu)

- $\sigma, i \models \pi$ iff $\pi \in \sigma_i$.
- $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$.
- $\sigma, i \models \phi_1 \wedge \phi_2$ iff $\sigma, i \models \phi_1$ and $\sigma, i \models \phi_2$.
- $\sigma, i \models \bigcirc\phi$ iff $i < h$ and $\sigma, i+1 \models \phi$.
- $\sigma, i \models \phi_1 \,\mathcal{U}\, \phi_2$ iff for some j such that $i \leq j \leq h$, we have that $\sigma, j \models \phi_2$, and for all $k, i \leq k < j$, we have that $\sigma, k \models \phi_1$.

An LTL$_f$ formula ϕ can be translated into a Nondeterministic Finite Automaton:

Definition 2.1: (NFA) A *Nondeterministic Finite Automaton* (NFA) $\mathcal{A}$ of an LTL$_f$ formula ϕ over $2^{\mathcal{AP}}$ is defined as a tuple $\mathcal{A}(\phi) = \left(\mathcal{Q}_\mathcal{A}, \mathcal{Q}^0_\mathcal{A}, \Sigma, \rightarrow_\mathcal{A}, \mathcal{Q}^F_\mathcal{A}\right)$, where

- $\mathcal{Q}_\mathcal{A}$ is the set of states;
- $\mathcal{Q}^0_\mathcal{A} \subseteq \mathcal{Q}_\mathcal{A}$ is a set of initial states;
- $\Sigma = 2^{\mathcal{AP}}$ is an alphabet;
- $\rightarrow_\mathcal{A} \subseteq \mathcal{Q}_\mathcal{A} \times \Sigma \times \mathcal{Q}_\mathcal{A}$ is the transition relation;
- $\mathcal{Q}^F_\mathcal{A} \subseteq \mathcal{Q}_\mathcal{A}$ is a set of accepting/final states.

A *finite run* $\rho_\mathcal{A}$ of $\mathcal{A}$ over a finite word $\sigma = \sigma_0\sigma_1 \ldots \sigma_h$, $\sigma_i = 2^{\mathcal{AP}}$, $\forall i \in [h]$, is a sequence $\rho_\mathcal{A} = q^0_\mathcal{A} q^1_\mathcal{A} \ldots q^{h+1}_\mathcal{A}$ such that $q^0_\mathcal{A} \in \mathcal{Q}^0_\mathcal{A}$ and $(q^i_\mathcal{A}, \sigma_i, q^{i+1}_\mathcal{A}) \in \rightarrow_\mathcal{A}$, $\forall i \in [h]$. A run $\rho_\mathcal{A}$ is called *accepting* if $q^{h+1}_\mathcal{A} \in \mathcal{Q}^F_\mathcal{A}$.

The dynamics of robot r is captured by a Transition System:

Definition 2.2: (TS) A *Transition System* for robot r is a tuple $\mathcal{T}(r) = \{\mathcal{S}_r, s^0_r, \rightarrow_r, \mathcal{AP}_r, \mathcal{L}_r\}$ where:

- $\mathcal{S}_r$ is the set of discrete states of robot r, and $s_r \in \mathcal{S}_r$ denotes a specific state;
- $s^0_r \in \mathcal{S}_r$ is the initial state of robot r;
- $\rightarrow_r \subseteq \mathcal{S}_r \times \mathcal{S}_r$ is the transition relation;
- $\mathcal{AP}_r$ is the set of atomic propositions related to robot r;
- $\mathcal{L}_r : \mathcal{S}_r \rightarrow 2^{\mathcal{AP}_r}$ is the observation (labeling) function that returns a subset of atomic propositions that are satisfied, i.e., $\mathcal{L}_r(s_r) \subseteq \mathcal{AP}_r$.

III. Hierarchical LTL$_f$

A. Syntax of H-LTL$_f$

Definition 3.1: (Hierarchical LTL$_f$) H-LTL$_f$ is structured into K levels, labeled $L_1, \ldots, L_K$, arranged from the highest to the lowest. Each level L_k with $k \in [K]_+$ contains n_k LTL$_f$ formulas. The H-LTL$_f$ specification is represented as $\Phi = \left\{\phi^i_k \mid k \in [K]_+, i \in [n_k]_+\right\}$, where ϕ^i_k denotes the i-th LTL$_f$ formula at level L_k. Let Φ_k denote the set of formulas at level L_k, and let $\texttt{Prop}(\phi^i_k)$ represent the set of propositions appearing in formula ϕ^i_k. The H-LTL$_f$ follows these rules:

1) There is exactly one formula at the highest level: $n_1 = 1$.
2) Each formula at level L_k consists either entirely of atomic propositions, i.e., $\texttt{Prop}(\phi^i_k) \subseteq \mathcal{AP}$, or entirely of formulas from the next lower level, i.e., $\texttt{Prop}(\phi^i_k) \subseteq \Phi_{k+1}$.
3) Each formula at level L_{k+1} appears in exactly one formula at the next higher level: $\phi^i_{k+1} \in \bigcup_{j \in [n_k]_+} \texttt{Prop}(\phi^j_k)$ and $\texttt{Prop}(\phi^{j_1}_k) \cap \texttt{Prop}(\phi^{j_2}_k) = \varnothing$, for $j_1, j_2 \in [n_k]_+$ and $j_1 \neq j_2$.

For a formula ϕ^i_k at a non-highest level, we slightly bend the notation to use ϕ^i_k to represent the same symbol at the higher level L_{k-1}, which we refer to as *composite proposition*. ϕ^i_k represents not only the i-th formula at level L_k, but also the corresponding composite proposition at level L_{k-1}. When ϕ^i_k appears at the right side in a certain formula, we consider it as a composite proposition. When it appears at the left side as a standalone formula, we refer to it as a specification.

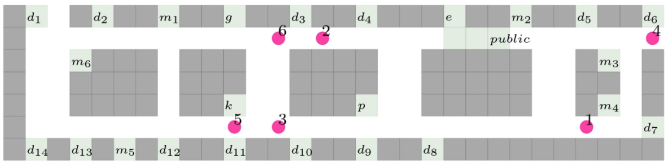

Fig. 1: The office building in a grid-based layout, where areas d_1 to d_{14} represent desks, m_1 to m_6 are meeting rooms, e stands for the elevator, g for the garbage room, p for the printer room, and k for the coffee kitchen. Areas marked as "public" indicate public spaces. Obstacles are illustrated in gray. The locations of robots are shown as numbered red dots.

Example 1: (H-LTL$_f$) We use the office environment described in [7] for service tasks, as depicted in Fig. 1. This 30×7 grid map features 14 desk areas, 6 meeting rooms, and several other functional rooms. Additionally, we distribute m robots at various locations.

Scenario 1: The task is that robots are required to distribute documents to desks d_{10}, d_7, and d_5, and avoid public areas while carrying the document. Let carry denote the action of the robot carrying the document. Note that the task does not assign specific robots. The standard LTL$_f$ specification is

$$\begin{aligned}\phi = &\Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_{10} \wedge \bigcirc\neg\mathsf{carry})) \\ &\wedge \Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_7 \wedge \bigcirc\neg\mathsf{carry})) \\ &\wedge \Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_5 \wedge \bigcirc\neg\mathsf{carry})) \\ &\wedge \Box(\mathsf{carry} \Rightarrow \neg\mathsf{public})\end{aligned} \tag{2}$$

The H-LTL$_f$ specifications are

$$\begin{aligned}L_1: \quad &\phi^1_1 = \Diamond\phi^1_2 \wedge \Diamond\phi^2_2 \wedge \Diamond\phi^3_2 \\ L_2: \quad &\phi^1_2 = \Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_{10} \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\ &\phi^2_2 = \Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_7 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\ &\phi^3_2 = \Diamond(p \wedge \mathsf{carry}\, \mathcal{U}\, (d_5 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\ &\mathsf{notpublic} := \Box(\mathsf{carry} \Rightarrow \neg\mathsf{public})\end{aligned} \tag{3}$$

There are two levels, L_1 and L_2, with L_1 having one formula and L_2 having three formulas. The symbol ϕ^1_2, appearing on the left side of the equal sign, is a specification at level L_2 but is a composite proposition at level L_1 since it is on the right side of the equal sign. $\Phi_1 = \{\phi^1_1\}$ and $\Phi_2 = \{\phi^1_2, \phi^2_2, \phi^3_2\}$, $\texttt{Prop}(\phi^1_1) = \{\phi^1_2, \phi^2_2, \phi^3_2\} \subseteq \Phi_2$ and $\texttt{Prop}(\phi^1_2) \subseteq \mathcal{AP}$.

Definition 3.2: (Specification hierarchy tree) The specification hierarchy tree, denoted as $\mathcal{G}_h = (\mathcal{V}_h, \mathcal{E}_h)$, is a tree where each node represents a specification within the H-LTL$_f$, and an edge (u, v) indicates that specification u contains specification v as a composite proposition. Any H-LTL$_f$ specifications can be turned into a specification hierarchy tree.

From a tree perspective, the level k of a specification node ϕ^i_k corresponds to its depth, defined as the number of nodes along the longest path from the root to that node.

Definition 3.3: (Leaf and Non-leaf Specifications) A specification is termed as a leaf specification if the associated node in the graph $\mathcal{G}_h$ does not have any children; otherwise, it is referred to as a non-leaf specification.

Let Φ_{leaf} denote the set of leaf specifications. Based on Def. 3.1, leaf specifications consist exclusively of atomic propositions, while non-leaf specifications consist solely of composite propositions.

B. Semantics of H-LTL$_f$

Given that H-LTL$_f$ includes multiple specifications, we consider multiple systems, particularly transition systems. At any given state, a system may be addressing a particular specification. We associate the state s_r of a system $\mathcal{T}(r)$ with a leaf specification $\psi_r \in \Phi_{\text{leaf}}$. This association establishes a state-specification pair (s_r, ψ_r), indicating that at state s_r, the system $\mathcal{T}(r)$ is engaged in satisfying the leaf specification ψ_r.

Definition 3.4: (State-Specification Sequence) A state-specification sequence with a horizon h, represented as τ, is a timed sequence $\tau = \tau_0\tau_1\tau_2\ldots\tau_h$. Here, $\tau_i = ((s_1^i, \psi_1^i), (s_2^i, \psi_2^i), \ldots, (s_N^i, \psi_N^i))$ is the collective state-specification pairs of N systems at the i-th timestep, where $s_r^i \in \mathcal{S}_r$, and $\psi_r^i \in \Phi_{\text{leaf}} \cup \{\epsilon\}$, with ϵ indicating the system's non-involvement in any leaf specification at that time.

Given the state-specification sequence τ and a leaf specification ϕ, we generate a word as $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n$, where $\sigma_i = \{\mathcal{L}_r(s_r^i) \mid (s_r^i, \psi_r^i) \in \tau_i \text{ and } \psi_r^i = \phi\}$, represents the collective observations generated by the systems engaging in the leaf specification ϕ at instant i. Furthermore, given a leaf (non-leaf) specification ϕ and an *input* word $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n$ with $\sigma_i \subseteq \texttt{Prop}(\phi)$, consisting of its atomic (composite) propositions, we construct an *output* word of ϕ, denoted by $\sigma' = \sigma'_0, \sigma'_1, \ldots, \sigma'_n$, where σ'_i is either $\varnothing$ or $\{\phi\}$, tracking the satisfaction of ϕ. Let $\sigma_{i,j} = \sigma_i, \ldots, \sigma_j$ denote the segment of the input word between indices i and j.

Definition 3.5: (Output word) Given a specification ϕ and an input word $\sigma = \sigma_0, \sigma_1, \ldots, \sigma_n$ with $\sigma_i \subseteq \texttt{Prop}(\phi)$, the output word $\sigma' = \sigma'_0, \sigma'_1, \ldots, \sigma'_n$ of ϕ is defined as:

- By default, $\sigma'_{-1} = \{\phi\}$.
- $\sigma'_j = \{\phi\}$ if $\sigma_{i+1,j}, i+1 \models \phi$, where i (with $i < j$) is the most recent instant when ϕ was satisfied, i.e., $i = max\{k \mid \sigma'_k = \{\phi\}, k = -1, 0, \ldots, j-1\}$.

The satisfaction relation $\models$ follows the same definition as introduced in Section II, treating composite propositions as atomic propositions for non-leaf specifications. The progress of ϕ resets after it is satisfied at instant i, and it is considered satisfied again at instant j if the input word segment $\sigma_{i+1,j}$ satisfies ϕ. In the context of robots, this implies that the duration of task ϕ extends from $i+1$ to j, with the completion time at instant j. The task's completion status is not persistent, meaning that once completed, it must start from scratch if repeated. The output words of specifications at level L_{k+1} serve as the input words and produce output words of specifications at level L_k. Thus, starting from the words of leaf specifications generated by the state-specification sequence τ and proceeding in a bottom-up manner, we ultimately obtain an output word of the root specification ϕ_1^1. The whole H-LTL$_f$ is satisfied by a state-specification sequence τ if the output word of ϕ_1^1 contains ϕ_1^1. Algorithmically, a post-order traversal can be used to traverse the specification hierarchy tree to compute the output word of ϕ_1^1.

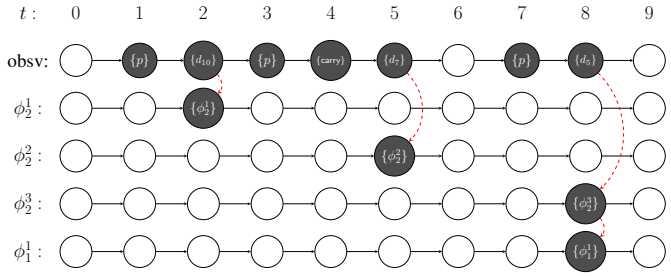

Fig. 2: Each row illustrates either the observation or the output word of a specification. The filled nodes indicate scenarios where an atomic or composite proposition is satisfied. The observations (obsv) generate the input word for leaf specifications $\phi_2^1 \sim \phi_2^3$, while the combination of output words for specifications $\phi_2^1 \sim \phi_2^3$ serve as the input word for the non-leaf specification ϕ_1^1. The dashed arrows illustrate the correspondence between specific inputs that lead to the satisfaction of a given specification.

Definition 3.6: (Semantics) Given H-LTL$_f$ specifications Φ and a state-specification sequence τ, the sequence τ satisfies Φ if the root specification ϕ_1^1 appears in the output word of ϕ_1^1.

Example 1: continued (Semantics) The output words for the specifications in (3) are illustrated in Fig. 2. □

IV. Problem Formulation

Given a state-specification sequence $\tau = \tau_0\tau_1\ldots\tau_h$ where $\tau_i = ((s_1^i, \psi_1^i), (s_2^i, \psi_2^i), \ldots, (s_N^i, \psi_N^i))$, the cost for robot r, such as energy consumption or completion time, is represented as $c_r = \sum_{i=0}^{h-1} c_r(s_r^i, s_r^{i+1})$, where $c_r(s_r^i, s_r^{i+1})$ denotes the cost incurred transitioning between states for robot r. The goal is to minimize the additive cost, expressed as

$$J(\tau) = \sum_{r=1}^{N} c_r \tag{4}$$

Finally, the problem can be formulated as follow:

Problem 1: Given transition systems for N robots and the H-LTL$_f$ specifications Φ, find an optimal state-specification sequence τ^* that satisfies Φ and minimizes $J(\tau^*)$.

We propose a search-based planning algorithm that enables simultaneous task allocation and action planning. The core idea is to approximate the global search space as a collection of loosely connected subspaces, where each subspace corresponds to an individual LTL$_f$ formula within the H-LTL$_f$. The planner conducts most of the search locally within a single subspace, transitioning to neighboring subspaces only when certain conditions—derived from the structure of the decomposed automata—are met. Due to space constraints, we omit technical details.

scenario	l_{std}	l_{hier}	$\mathcal{A}_{\text{std}}$	$\mathcal{A}_{\text{hier}}$	t_{std}	t_{hier}	c_{std}	c_{hier}
1	**18**	19	(17, 39)	**(12, 13)**	14.1±3.7	**4.9±2.3**	71.0±6.3	**69.0±5.7**
2	**24**	35	(56, 326)	**(20, 31)**	39.6±2.5	**7.0±3.1**	90.4±4.0	**88.6±7.0**
3	**35**	52	(180, 1749)	**(30, 49)**	timeout	**14.5±4.1**	—	**97.1±5.9**
$1 \wedge 2$	**43**	57	(868, 12654)	**(36, 49)**	timeout	**16.4±7.1**	—	**148.8±8.4**
$1 \wedge 3$	**54**	74	(2555, 69858)	**(46, 67)**	timeout	**47.9±22.3**	—	**166.4±8.0**
$2 \wedge 3$	**60**	90	(6056, 325745)	**(54, 85)**	timeout	**42.5±16.9**	—	**175.4±6.9**
$1 \wedge 2 \wedge 3$	**79**	111	—	**(70, 112)**	timeout	**89.6±28.1**	—	**246.6±8.3**
$(1 \vee 2) \wedge 3$	**79**	110	—	**(66, 98)**	timeout	**71.5±17.6**	—	**213.1±23.4**
$1 \vee 2 \vee 3$	**79**	109	—	**(64, 94)**	timeout	**63.4±20.9**	—	**163.9±40.5**

TABLE I: The comparative analysis focuses on two different types of LTL$_f$ specifications. We denote the lengths of the standard and H-LTL$_f$ specifications as l_{std} and l_{hier}, respectively. The sizes of the corresponding NFAs are represented by $\mathcal{A}_{\text{std}}$ and $\mathcal{A}_{\text{hier}}$, which detail the number of nodes and edges, with the node count listed first. In terms of solutions, the runtimes for the standard and H-LTL$_f$ specifications are indicated by t_{std} and t_{hier}, respectively. Additionally, the plan horizons, or the lengths of the solutions, for the standard and hierarchical specifications are denoted by c_{std} and c_{hier}, respectively.

V. Simulation Experiments

A. Scenarios

1) Scenario 2: Transport the paper bin from desk d_5 to area g for emptying, avoiding the public area while it is full. Return an empty bin from g to desk d_5. The atomic propositions are: default: the robot is not carrying any object, carrybin: the robot is carrying a full paper bin, dispose: the robot is disposing of garbage, emptybin: the robot is carrying an empty paper bin, and public: the robot is located in a public area.

$$
\begin{aligned}
L_1:\ & \phi_1^1 = \Diamond\phi_2^1 \wedge \Diamond\phi_2^2 \\
L_2:\ & \phi_2^1 = \Diamond(d_5 \wedge \mathsf{default} \wedge \bigcirc((\mathsf{carrybin}\ \mathcal{U}\ \mathsf{dispose}) \wedge \Diamond\mathsf{default})) \\
& \qquad \wedge \Box(\mathsf{carrybin} \Rightarrow \neg\mathsf{public}) \\
& \phi_2^2 = \Diamond(g \wedge \bigcirc(g \wedge \mathsf{emptybin}) \wedge \Diamond(d_5 \wedge \bigcirc(d_5 \wedge \mathsf{default})))
\end{aligned}
$$

2) Scenario 3: Take a photo in meeting rooms m_1, m_4, and m_6. The camera should be turned off for privacy reasons when not in meeting rooms. Deliver a document from desk d_5 to d_3, ensuring it does not pass through any public areas, as the document is internal and confidential. Guide a person waiting at desk d_{11} to meeting room m_6. Let guide, photo, and camera denote the actions of the robot guiding a person, capturing a photo, and activating the camera, respectively.

$$
\begin{aligned}
L_1:\ & \phi_1^1 = \Diamond\phi_2^1 \wedge \Diamond\phi_2^2 \wedge \Diamond\phi_2^3 \\
L_2:\ & \phi_2^1 = \Diamond\phi_3^1 \wedge \Diamond\phi_3^2 \wedge \Diamond\phi_3^3 \\
& \phi_2^2 = \Diamond(d_5 \wedge \mathsf{carry}\ \mathcal{U}\ (d_3 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\
& \phi_2^3 = \Diamond(d_{11} \wedge \mathsf{guide}\ \mathcal{U}\ (m_6 \wedge \bigcirc\neg\mathsf{guide})) \\
L_3:\ & \phi_3^1 = \Diamond(m_1 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera}) \\
& \phi_3^2 = \Diamond(m_4 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera}) \\
& \phi_3^3 = \Diamond(m_6 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera}) \\
& \mathsf{meeting} := m_1 \vee m_2 \vee m_3 \vee m_4 \vee m_5 \vee m_6
\end{aligned}
$$

3) Combinations of scenarios 1, 2 and 3: We examine combinations of any two of these tasks as well as the combination of all three. Due to space constraints, we only detail the scenario involving the final occurrence of all three tasks.

$$
\begin{aligned}
L_1:\ & \phi_1^1 = \Diamond\phi_2^1 \wedge \Diamond\phi_2^2 \wedge \Diamond\phi_2^3 \\
L_2:\ & \phi_2^1 = \Diamond\phi_3^1 \wedge \Diamond\phi_3^2 && \text{(scenario 2)} \\
& \phi_2^2 = \Diamond\phi_3^3 \wedge \Diamond\phi_3^4 \wedge \Diamond\phi_3^5 && \text{(scenario 1)} \\
& \phi_2^3 = \Diamond\phi_3^6 \wedge \Diamond\phi_3^7 \wedge \Diamond\phi_3^8 && \text{(scenario 3)}
\end{aligned}
$$

$$
\begin{aligned}
L_3:\ & \phi_3^1 = \Diamond(d_5 \wedge \mathsf{default} \wedge \bigcirc((\mathsf{carrybin}\ \mathcal{U}\ \mathsf{dispose}) \wedge \Diamond\mathsf{default})) \\
& \qquad \wedge \Box(\mathsf{carrybin} \Rightarrow \neg\mathsf{public}) \\
& \phi_3^2 = \Diamond(g \wedge \bigcirc(g \wedge \mathsf{emptybin}) \wedge \Diamond(d_5 \wedge \bigcirc(d_5 \wedge \mathsf{default}))) \\
& \phi_3^3 = \Diamond(p \wedge \mathsf{carry}\ \mathcal{U}\ (d_{10} \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\
& \phi_3^4 = \Diamond(p \wedge \mathsf{carry}\ \mathcal{U}\ (d_7 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\
& \phi_3^5 = \Diamond(p \wedge \mathsf{carry}\ \mathcal{U}\ (d_5 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\
& \phi_3^6 = \Diamond\phi_4^1 \wedge \Diamond\phi_4^2 \wedge \Diamond\phi_4^3 \\
& \phi_3^7 = \Diamond(d_5 \wedge \mathsf{carry}\ \mathcal{U}\ (d_3 \wedge \bigcirc\neg\mathsf{carry})) \wedge \mathsf{notpublic} \\
& \phi_3^8 = \Diamond(d_{11} \wedge \mathsf{guide}\ \mathcal{U}\ (m_6 \wedge \bigcirc\neg\mathsf{guide})) \\
L_4:\ & \phi_4^1 = \Diamond(m_1 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera}) \\
& \phi_4^2 = \Diamond(m_4 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera}) \\
& \phi_4^3 = \Diamond(m_6 \wedge \mathsf{photo}) \wedge \Box(\neg\mathsf{meeting} \Rightarrow \neg\mathsf{camera})
\end{aligned} \tag{5}
$$

If the objective is to accomplish either scenario, this can be indicated by replacing the formula at level L_1 with $\phi_1^1 = \Diamond(\phi_2^1 \vee \phi_2^2 \vee \phi_2^3)$. Furthermore, the expression $\phi_1^1 = \Diamond(\phi_2^1 \vee \phi_2^2) \wedge \Diamond\phi_2^3$ specifies that either task 1 or task 2 must be completed, in addition to task 3.

B. Comparison with Existing Works

We use $m = 6$ robots and compare our method with the approach in [7]. Robot locations in each scenario are randomly assigned within the free space. The performance, in terms of average runtimes and costs over 20 runs, is detailed in Tab. I and includes the length of formulas and sizes of automata. The length of a formula is the total number of logical and temporal operators. Upon reviewing the results, [7]'s method failed to generate solutions for the last seven tasks within the one-hour limit. For tasks 1 and 2, our method produced solutions more quickly and with comparable costs. The failure of [7]'s approach is attributed to the excessively large automata it generates, sometimes with hundreds of thousands of edges, e.g., 325745 edges for scenario $2 \wedge 3$, making the computation of the decomposition set time-consuming as it requires iterating over all possible runs. For the most complex scenario, such as $1 \wedge 2 \wedge 3$, generating an automaton within one hour is impossible. In contrast, our method was able to find a solution in around 90 seconds. Moreover, considering the last three tasks, which require completing all, two, or only one task, both the runtime and cost decrease as the number of required tasks is reduced.

REFERENCES

[1] Jim Woodcock, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. Formal methods: Practice and experience. ACM computing surveys (CSUR), 41(4):1–36, 2009.

[2] Vince Kurtz and Hai Lin. Temporal logic motion planning with convex optimization via graphs of convex sets. IEEE Transactions on Robotics, 2023.

[3] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. Science, 331(6022):1279–1285, 2011.

[4] Charles Kemp, Andrew Perfors, and Joshua B Tenenbaum. Learning overhypotheses with hierarchical bayesian models. Developmental Science, 10(3):307–321, 2007.

[5] Giuseppe De Giacomo and Moshe Y Vardi. Linear temporal logic and linear dynamic logic on finite traces. In Twenty-Third International Joint Conference on Artificial Intelligence, 2013.

[6] Christel Baier and Joost-Pieter Katoen. Principles of model checking. MIT press Cambridge, 2008.

[7] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. The International Journal of Robotics Research, 37(7):818–838, 2018.