2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Active Inference for Reactive Temporal Logic Motion Planning

Ziyang Chen, Zhangli Zhou, Lin Li, and Zhen Kan

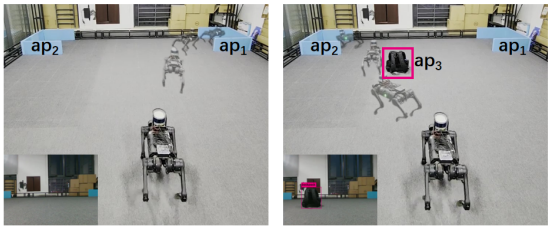
2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10611484

***Abstract*— Reactive planning enables the robots to deal with dynamic events in uncertain environments. However, existing methods heavily rely on the predefined hard-coded robot behaviors, e.g, a pre-coded temporal logic formula that specifies how robot should react. Little attention has been paid for autonomous generation of reactive tasks specifications during the runtime. As a first attempt towards this goal, this work develops a real-time decision-making and motion planning framework. It allows the robot to follow a global task planned offline while taking proactive decisions and generating temporal logic specifications for local reactive tasks when encountering dynamic events. Specifically, inspired by the causal knowledge graph, a proposition graph is developed, based on which the decision module encode the environment and the task as the Boolean logic and linear temporal logic (LTL), respectively. Based on the established proposition graph and perceived environment, the agent can autonomously generate an LTL formula to realize the local temporary task. A joint sampling algorithm is then developed, in which the automaton states of local and global task are jointly considered to generate a feasible planning that satisfies both global and local tasks. Experiments demonstrate the effectiveness of the proposed decision-making and motion planning.**

I. INTRODUCTION

Active inference, a capability of reasoning and planning, is crucial for online robot control, especially when operating in dynamic environments. To deal with unexpected and dynamic events, reactive planning with temporal logic specifications has shown promising results [1], [2]. Nevertheless, these approaches heavily rely on predefined hard-coded robot behaviors, e.g, a pre-coded temporal logic formula that specifies how robot should react [3]–[5]. Little attention has been paid to active inference to enable intelligent reasoning and decision-making during the runtime. In practice, the robot is often desired to take proactive decisions at run-time for local reactive tasks without violating global tasks.

As a motivating example, a quadruped robot is assigned a global security task of repeatedly checking areas of interest, as shown in in Fig. 1. If a suspicious object is found, a local temporary task can be triggered, which requires the robot to deliver the suspicious object to area 2. In terms of temporal logic specifications, the global and local tasks can be written as $\Phi_G = GFap_1 \wedge GFap_2$ and $\Phi_L = F(ap_3 \wedge X(Fap_2))$, respectively. In conventional approaches, these tasks have to be hard-coded in advance (i.e., specify exactly what actions should be taken if which object is detected). This can be tedious and challenging, as it is hard to enumerate all potential suspicious objects and the corresponding actions. It is highly desired that the robot is only given the order of "delivering any suspicious objects to area 2" and the robot can infer suspicious objects and generate the corresponding temporal logic formula that specifies the local task on its own using active inference based on the observed objects and common sense. However, few methods in literature can identify the reactive task expressed by temporal logic autonomously and proactively determine which should be done subsequently. The lack of proactive decision making significantly limits their applicability and flexibility, especially in dynamic environments. Hence, this work is particularly motivated to proactive determination and generation of local reactive tasks using active inference.



(a) (b)

Fig. 1. (a) The robot performs the global task of visiting area 1 (ap_1) and area 2 (ap_2) in sequence. (b) The robot detects a suspicious backpack (ap_3) and delivers it to area 2 (ap_2).

Related Work. Due to its rich expressivity and resemblance to natural language, linear temporal logic (LTL) has been widely used to describe complex robotic tasks and specify robot behaviors [6]–[9]. However, LTL formulas in most applications are still required to be handcrafted in advance or generated by large number of trajectories [10], [11]. To relax this limitation, a knowledge base is often preferred. The causal knowledge graph is employed to facilitate decision making [12]–[14], which enables agents to determine subsequent actions according to the observed environment and task requirement. However, complex tasks, such as LTL tasks, are usually too complex to be represented by independent nodes and thus the causal knowledge graph is hard to be directly applied.

Furthermore, reactive planning in the complex environment is also a challenge. Recent reactive planning generally focuses on dynamic environment [15], independent sub-task, e.g. local temporary task [16], [17], or infeasible sub-tasks [18]. When encountering a local temporary task during the execution of a global task, it is necessary to know the progress of the current task via automaton-based approaches [4], [19]. Among them, sampling-based methods [3], [20]

Z. Chen, Z. Zhou, L. Li, and Z. Kan (Corresponding Author) are with the Department of Automation at the University of Science and Technology of China, Hefei, Anhui, China, 230026.

This work was supported in part by the National Natural Science Foundation of China under Grant U2013601 and Grant 62173314.

show great potentials in terms of comprehensive applicability and faster solution time, which makes such methods good candidate for complex tasks in real-time [21]. However, when the local task is coupled with the global task, in addition to the progress of the global task, we should also consider the conflict between the local and global tasks. Attempts to address this issue include the multi-task transition planning method in [22] and the method of combining multiple tasks in [23].

To address the above challenges, this work develops a real-time decision-making and motion planning framework that allows the robot to follow a global task planned offline while taking proactive decisions and generating temporal logic specifications for local reactive tasks when encountering dynamic events. To enable proactive decision making for local temporary tasks, inspired by the causal knowledge graph [24], a novel proposition graph is developed in this work to describe the relationships of sub-tasks. Specifically, the environment information and the task information are described by Boolean logic and temporal logic based on the propositional graph. Based on the automaton sampling method in [3], a joint sampling algorithm is then developed. Through the sampling of atomic propositions, the automaton states of local and global task are jointly considered to generate a feasible planning that satisfies both global and local tasks.

The main contributions are summarized as follows. To the best of our knowledge, this is one of the first attempts to leverage the proposition graph to establish the temporal and logical representation between sensory input (i.e., observed objects and environment) and desired tasks. Based on the established proposition graph and perceived environment via images or point clouds, the agent can autonomously generate an LTL formula to realize the local temporary task, rather than using a pre-coded LTL formula as in many existing works. To deal with the multi-task reactive planning, the joint sampling planner is developed, in which the automaton states of local and global task are jointly considered to generate a feasible planning that satisfies both global and local tasks. Experiments demonstrate the effectiveness of the proposed decision-making and motion planning.

II. PRELIMINARIES

LTL is a formal language defined over a set of atomic propositions AP with Boolean and temporal operators. The syntax of LTL is defined as:

$$\phi := true | ap | \phi_1 \wedge \phi_2 | \neg \phi_1 | X\phi | \phi_1 U \phi_2$$

where $ap \in AP$ is an atomic proposition, *true*, $\neg$ (negation), and $\wedge$ (conjunction) are propositional logic operators, and X (next) and U (until) are temporal operators. Other propositional logic operators such as *false*, $\vee$ (disjunction), and temporal operators such as G (always) and F (eventually) can also be defined [25].

The word $\boldsymbol{\pi} = \pi_0 \pi_1 \dots$ is an infinite sequence where $\pi_i \in 2^{AP}$, $\forall i \in \mathbb{Z}_{\geq 0}$, with 2^{AP} representing the power set of AP. Given a word $\boldsymbol{\pi}$, denote by $\boldsymbol{\pi}[j \dots] = \pi_j \pi_{j+1} \dots$ and $\boldsymbol{\pi}[\dots j] = \pi_0 \dots \pi_j$.

More details about LTL syntax, semantics, and model checking are referred to [25]. An LTL formula can be converted to a Non-deterministic Büchi Automaton (NBA).

Definition 1. An NBA is a tuple $B = \{S, S_0, \Sigma, \delta, F_S\}$, where S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, $\Sigma = 2^{AP}$ is the finite alphabet, $\delta \subseteq S \times \Sigma \times S$ is the state transition, and $F_S \subseteq S$ is the set of accepting states.

Let $\Delta : S \times S \to 2^{\Sigma}$ denote the set of atomic propositions that enables state transitions in NBA, i.e., $\forall \pi \in \Delta(s, s')$, there exists $(s, \pi, s') \in \delta$. Then, the NBA can also be defined as $B = \{S, S_0, \Sigma, \Delta, F_S\}$. A valid run $\boldsymbol{s} = s_0 s_1 s_2 \dots$ of B_ϕ generated by the word $\boldsymbol{\pi}$ with $\pi_i \in \Delta(s_{i-1}, s_i)$, $\forall i \in \mathbb{N}_{\geq 1}$, is called accepting, if $\boldsymbol{s}$ intersects with F_S infinite often. An LTL formula can be translated to an NBA by the tool [26]. In this paper, NBA will be used to track the progress of the satisfaction of LTL tasks.

III. PROBLEM FORMULATION

Consider a bounded workspace $M \subset \mathbb{R}^2$, where a position p is denoted as $p = (x, y) \in M$. Suppose M consists of a number of non-overlapped areas of interest M_I and a free space M_U, i.e., $M = M_I \cup M_U, M_I \cap M_U = \emptyset$. Let AP represent the set of tasks that can be performed in M. The label function $L : M \to AP$ maps p to the corresponding atomic proposition $ap \in AP$, i.e., $L(p) = ap$. We further denote by $LM : AP \to M$ to indicate the position p where an atomic proposition $ap \in AP$ is executable. The robot operating in M is defined as a tuple $R = \{p_0, p, v, \Omega\}$, where p_0 is the initial position, p is the current position, v is the maximum linear velocity, and Ω is the robot observation.

Definition 2. The robot observation is a tuple $\Omega = \{O, O_M, O_P\}$, where $O = \{o_1, o_2, \dots, o_{n_o}\}$ is the set of observed objects with $o_i \in O$ representing the ith object and n_o representing the total number of observed objects, $O_M : O \to M$ maps an object to its position in M, and $O_P : O \to AP$ maps an object to its corresponding atomic proposition.

Consider a global LTL task Φ_G and the corresponding NBA is denoted by $B_G = (S_G, S_{G0}, \Sigma_G, \Delta_G, F_G)$.

Definition 3. (Global Plan): The global plan Π corresponding to Φ_G is defined as $\Pi = (\boldsymbol{s}_G, \boldsymbol{p}, \boldsymbol{\pi})$, where $\boldsymbol{s}_G = s_0^G s_1^G s_2^G \dots$ is the sequence of states of B_G, $\boldsymbol{p} = p_0 p_1 p_2 \dots$ is the sequence of execution positions with p_0 indicating the initial position of the robot, and $\boldsymbol{\pi} = \pi_0 \pi_1 \pi_2 \dots$ is sequence of sub-tasks with π_0 indicating the empty task.

In Def. 3, $s_i^G \in S_G$ indicates the state of B_G after the completion of π_i, i.e., $\pi_i \in \Delta(s_{i-1}^G, s_i^G)$, and $p_i \in M$ indicates the task execution location of π_i, i.e., $p_i = LM(\pi_i)$. By defining $\Pi_i = (s_i^G, p_i, \pi_i)$, the global planning can be written as $\Pi = \Pi_0 \Pi_1 \Pi_2 \dots$. We denote by $\Pi \models \Phi_G$ if the planning Π satisfies the LTL formula Φ_G. Based on the prefix-suffix structure, the planning Π can be further written

in the form of $\Pi = \Pi_{pre}\Pi_{tra}\Pi_{suf}\Pi_{suf}...$, where Π_{pre} and Π_{suf} are finite prefix and finite cyclic suffix, respectively. As the end state of prefix and start state of suffix may be different, the finite transition Π_{tra} is developed to connect them [27]. Since $\Pi_{pre}\Pi_{tra}\Pi_{suf} \models \Phi$ also indicates that $\Pi_{pre}\Pi_{tra}\Pi_{suf}\Pi_{suf}... \models \Phi$, we only need to determine Π_{pre}, Π_{tra}, and Π_{suf} in Π.

When performing the global task Φ_G, the robot may encounter local temporary task based on the observation Ω. Such tasks are considered as local tasks Φ_L in this work. The local tasks Φ_L are specified by co-safe LTL formulas [28] to represent local temporary tasks which can be satisfied by finite-length prefix words. The co-safe LTL can be converted to a non-deterministic finite automata (NFA) [29], which is similar with the NBA.

Definition 4. (Local Plan): Given an NFA $B_L = (S_L, S_{L0}, \Sigma_L, \Delta_L, F_L)$ corresponding to Φ_L, the local plan is defined as $\Pi_{tem} = (\boldsymbol{s}_G, \boldsymbol{s}_L, \boldsymbol{p}, \boldsymbol{\pi})$, where $\boldsymbol{s}_G$, $\boldsymbol{p}$, and $\boldsymbol{\pi}$ are the same as the global plan Π and $\boldsymbol{s}_L = s_0^L s_1^L s_2^L \dots$ is the sequence of states of B_L.

Let AP_G and AP_L denote the atomic propositions of Φ_G and Φ_L, respectively. Since local tasks should also respect the global tasks, we denote by $\Pi_{tem} \models (\Phi_G, \Phi_L)$, if $\forall s_i^L \in \boldsymbol{s}_L$, it holds that $\pi_i \in \Delta_L(s_{i-1}^L, s_i^L)$ or $\pi_i \notin AP_L$, and $\forall s_i^G \in \boldsymbol{s}_G$, it holds that $\pi_i \in \Delta_G(s_{i-1}^G, s_i^G)$ or $\pi_i \notin AP_G$.

In a partially unknown environment, the agent has to operate based on its local observation of the environment. We hope that the agent can generate local task formula based on active inference of the observed environment, and obtain a plan that satisfies both local and global tasks. Then, the problem of autonomous reactive planning can be defined as follows.

Problem 1. Given the map M and the global task Φ_G, how to obtain the enforceable local tasks and their specifications Φ_L according to the observation Ω. Then, generate the new planning scheme Π_{tem}, Π_{pre}, Π_{tra}, Π_{suf}, which satisfy $\Pi_{tem} \models (\Phi_G, \Phi_L)$ and $\Pi_{pre}\Pi_{tra}\Pi_{suf} \models \Phi_G$.

IV. Proposition Graph Based Planning

To enable proactive decision making, the key idea is to generate task specifications for local tasks via active inference based on the knowledge base and the observed environment information Ω during runtime. The joint sampling is then developed for the planning of both global and local tasks. The overview of active inference based reactive planning is shown in Fig. 2.

A. Proposition Graph

Causal knowledge graph has been widely used to represent the causal relationship of entities such as $'cause'$, $'effect'$, and $'state'$. In [14], the causal knowledge graph is defined as a tuple $KG = \{E_k, R_k, F_k\}$, where E_k is a set of entities, $R_k = \{'cause', 'effect', 'state'\}$ is a set of relationships, and F_k is a set of facts where each fact is represented by a tuple $(e_h, r_k, e_t) \in F_k$ with $e_h \in E_k$, $e_t \in E_k$, and $r_k \in R_k$ representing the head entity, the tail entity, and the relation between e_h and e_t, respectively.

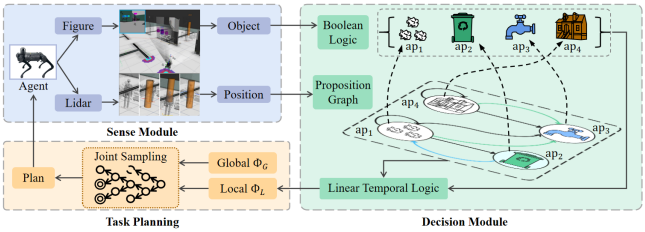


Fig. 2. Our approach consists of the sensing module, decision module, and the planning module. In the sensing module, the environment information Ω, such as the location of the object and the object classification, is collected via visual and Lidar feedback. In the decision module, the proposition graph PG is developed as a knowledge base to indicate the relationships between atomic propositions, and then is applied to generate the local task specification Φ_L based on Ω. Given the observed Ω, Φ_L and PG, the real-time planning via the joint sampling is developed in the task planning module to ensure the satisfaction of both Φ_G and Φ_L.

Motivated by the causal knowledge graph KG, we develop a proposition graph in this work consisting of atomic propositions with their associated attributes. The proposition graph contains the domain knowledge and can be applied to generate the temporal logic formula.

Definition 5. (**Proposition Graph**) The proposition graph is a tuple $PG = \{AP, R_p, F_p\}$, where the nodes represent the atomic propositions AP, $R_p = \{'before', 'after', 'related'\}$ represents the temporal relationship between atomic propositions, and F_p is the set of facts where each fact is a tuple $(ap_h, r, ap_t) \in F_p$ with $ap_h \in AP$, $ap_t \in AP$, and $r \in R_p$ represent the head proposition, the tail proposition, and the relationship between ap_h and ap_t, respectively.

For instance, if given a tuple $(ap_h, 'before', ap_t) \in F_p$, then ap_t should be executed before ap_h. If given a tuple $(ap_h, 'after', ap_t) \in F_p$, then ap_t should not be executed before ap_h. If $(ap_h, 'related', ap_t) \in F_p$, then the execution of ap_h in the environment indicates that ap_t should also be executed. To conveniently specify the relationship between atomic propositions, we define three functions $bf, af, re : AP \to 2^{AP}$, where $bf(ap_i) \subseteq AP$ indicates the set of preceding propositions such that $(ap_i, 'before', ap_j) \in F_p$ for all $ap_j \in bf(ap_i)$, $af(ap_i) \subseteq AP$ indicates the set of post-propositions such that $(ap_i, 'after', ap_j) \in F_p$ for all $ap_j \in af(ap_i)$, and $re(ap_i) \subseteq AP$ indicates the set of related propositions such that $(ap_i, 'related', ap_j) \in F_p$ for all $ap_j \in re(ap_i)$. The relationships in proposition graph can be obtained by existing causal knowledge graph [14] or LLM [30] to avoid manual pre-definition of robot behaviors.

B. Proactive Decision-Making Module

To enable proactive decision making for reactive local tasks, we first establish the propositional graph PG to represent the relationship of atomic propositions in the environment. The following example sheds light on how PG is constructed.

Example 1. In the right plot of Fig. 2, there are four objects in the environment: waste, trash bin, water faucet, and home, which correspond to four atomic propositions: picking up the waste by ap_1, throwing the waste into the trash bin by ap_2, washing hands by ap_3, and going home by ap_4. The knowledge base that specifies a set of rules is defined as follows: 1) picking up the waste before throwing it into the trash bin; 2) washing hands after entering home; and 3) washing hands after picking up the waste; 4) executing ap_1 and ap_4 near the corresponding objects; 5) picking up the waste is related to washing hands; (6) trashing the waste is related to picking up waste; 7) going home is related to washing hands. The corresponding PG is then constructed as in Fig. 2. The black, green, and blue arrows indicate the relationship of $'related'$, $'after'$, and $'before'$, respectively.

As outlined in Alg. 1, the decision module takes as input the local observation Ω, and based on the the established proposition graph PG, outputs the generated logical representations of the sense Φ_{in} and the local task Φ_{out}. Specifically, the Boolean logic representation of the sense Φ_{in} and its involved propositions AP_{in} are obtained by the observation Ω (lines 1-8). For the object $o \in O$, if there exists corresponding atomic proposition ap satisfying $O_p(o) = ap$, i.e., ap is in the environment, then add ap into the set AP_{in} and the logical representation of the environment Φ_{in}. Besides, $LM(ap)$ is set as the location $O_M(o)$ by the sense. The task formula Φ_{out} can be obtained by the formula generation function Get_Formula. Finally, Φ_{in} is returned as a logical representation of the sense, and Φ_{out} is returned as a logical representation of the local task to guide the task planner.

Algorithm 1: Proactive Decision Making

Input: PG, Ω
Output: Φ_{in}, Φ_{out}
1 Initialize $\Phi_{in} =' true'$, $AP_{in} = \emptyset$;
2 **for** *o in O* **do**
3 **if** $\exists ap \in AP = O_P(o)$ **then**
4 $\Phi_{in} = \Phi_{in} \wedge ap$;
5 Add ap to AP_{in};
6 $LM(ap) = O_M(o)$;
7 **end**
8 **end**
9 Initialize $\Phi_{out} =' true'$;
10 $\Phi_{out} = \mathsf{Get\_Formula}(AP_{in}, PG, \Phi_{out})$;
11 Return Φ_{in}, Φ_{out};

One ultimate goal of Alg. 1 is to autonomously generate an LTL formula for the encountered local task based on the observed environment. To achieve this goal, the function Get_Formula is developed to obtain the task formula iteratively based on PG, which is outlined in Alg. 2. Firstly, the sub-formula Φ_{sub} of the related propositions AP_r is constructed (Lines 1-12). For $ap_i \in AP_r$, if there exists $ap_j \in re(ap_i)$, then we add sub-formula Fap_j into Φ_{sub}, i.e., execute ap_j in the local task. Besides, if there exists temporal order, i.e. $bf(ap_i) \neq \emptyset$, $af(ap_i) \neq \emptyset$, we add the sub-formula $'((\neg ap_i)U(ap_j))'$ and $'((\neg ap_j)U(ap_i))'$ respectively, for the $'before'$ and $'after'$ relationships. Then, add Φ_{sub} into Φ_{out} as the new task formula. Secondly (Lines 14-17), get the new related atomic propositions by the current related propositions AP_r and add them into the set AP_{new}. Finally (Lines 18-23), if there is no ap related to the AP_r, then return Φ_{out}. Otherwise, continue to get new sub-formula iteratively according to the new set of related propositions AP_{new}, the proposition graph PG, and the currently generated task formula Φ_{out}.

Algorithm 2: Get_Formula

Input: AP_r, PG, Φ_{out}
Output: Φ_{out}
1 Initialize $\Phi_{sub} =' true'$;
2 **for** *ap_i in AP_r* **do**
3 **for** *ap_j in $re(ap_i)$* **do**
4 $\Phi_{sub} = \Phi_{sub} +' \wedge Fap_j'$;
5 **end**
6 **for** *ap_j in $bf(ap_i)$* **do**
7 $\Phi_{sub} = \Phi_{sub} +' \wedge((\neg ap_i)U(ap_j))'$;
8 **end**
9 **for** *ap_j in $af(ap_i)$* **do**
10 $\Phi_{sub} = \Phi_{sub} +' \wedge((\neg ap_j)U(ap_i))'$
11 **end**
12 **end**
13 $\Phi_{out} = \Phi_{out} + \Phi_{sub}$;
14 $AP_{new} = \emptyset$;
15 **for** *ap in AP_r* **do**
16 Add $re(ap)$ to AP_{new};
17 **end**
18 **if** $AP_{new} = \emptyset$ **then**
19 Return Φ_{out};
20 **else**
21 $\Phi_{out} = \mathsf{Get\_Formula}(AP_{new}, PG, \Phi_{out})$;
22 Return Φ_{out};
23 **end**

C. Task Planning Module

After constructing the local task, the robot is desired to complete the local task without violating the global task. As directly integrating the local formula into the global task will form a new formula, the current state of global task s_G can not be reflect in the new automata converted by the new formula. Therefore, B_G should be retained and combined with B_L for joint planning. To achieve this goal, inspired by the fast task planner [3], a joint sampling method is developed in this work. The idea behind of the joint sampling is to incrementally construct a joint tree $\mathcal{T}$ in real time and then search over it for task planning. Specifically, the joint tree is defined as a tuple $\mathcal{T} = (N, E, C)$, where N represents the set of nodes, E represents a set of transitions, and $C : N \to \mathbb{R}^+$ is the cost function indicating the cost of reaching the current node from the root n_0. Each node $n_i \in N$ is defined as $n_i = (ap_i, s_i^{GT}, s_i^{LT}) \in AP \times B_G \times B_L$, which indicates the current sub-task, local and global automaton states, respectively. A transition $(n_i, n_j) \in E$ is feasible, if it satisfies the local task Φ_L and global task Φ_G, i.e., it holds that $ap_j \in \Delta_L(s_i^{LT}, s_j^{LT})$ or $ap_j \notin AP_L$, and, $ap_j \in \Delta_G(s_i^{GT}, s_j^{GT})$ or $ap_j \notin AP_G$. The set E captures

feasible transitions among the nodes in N that satisfy both local task Φ_L and global task Φ_G.

As outlined in Alg. 3, we first generate the NBA B_G and NFA B_L corresponding to Φ_G and Φ_L, respectively, and initialize the joint tree with the root $N = \{n_0\}$ where $s_0^{GT} \in S_G$ is the current global task state and s_0^{LT} is the initial state of local task, i.e., $s_0^{LT} \in S_{L0}$. Then, the tree $\mathcal{T}$ grows from $\{n_0\}$ by sampling the atomic propositions. Suppose n_j is sampled as the parent node and ap_i is the next atomic proposition to be executed. Then, if ap_i is coupled with the local task Φ_L, we sample new local task state s_i^{LT} satisfying $ap_i \in \Delta(s_j^{LT}, s_i^{LT})$. Otherwise, s_i^{LT} is set as s_j^{LT}. The same procedure applies to the global task state s_i^{GT}. The pair (n_j, n_i) is considered as a new feasible transition and added to E. The cost value of n_i is determined by $C(n_i) = C(n_j) + \|LM(ap_i) - LM(ap_j)\|$.

Similar to the pruning approach in [3], the function Pruning in Alg. 3 is developed to remove redundant nodes to improve the path search efficiency. For instance, given a transition $(n_j, n_i) \in E$, if there exists a node $n_k \in N$ such that (n_k, n_i) is also a feasible transition and satisfies $C(n_k) + \|LM(ap_i) - LM(ap_k)\| < C(n_i)$, then we delete (n_j, n_i) from E, add (n_k, n_i) into E instead and set $C(n_i)$ as $C(n_k) + \|LM(ap_i) - LM(ap_k)\|$. If there exist nodes n_i and n_j satisfying $ap_i = ap_j$, $s_i^{LT} = s_j^{LT}$, $s_i^{GT} = s_j^{GT}$, $C(n_i) \leq C(n_j)$, then we delete n_j from N.

After constructing the tree $\mathcal{T}$, we select the node n_{min} that has the minimum cost among all nodes that complete the local task and F_G is reachable from its global state[1]. According to the node path from n_0 to n_{min}, for n_i in the path, we can construct local plan tuple $(s_i^{GT}, s_i^{LT}, LM(ap_i), ap_i)$ and combine them as the local plan Π_{tem}. According to the end state of local plan, we initialize the root node for global task and obtain the next global plan $\Pi_{pre}, \Pi_{tra}, \Pi_{suf}$ though Sampling method in [3].

Theorem 1. *The joint sampling in Alg. 3 is valid, i.e., if it finds a feasible planning scheme, it is guaranteed to satisfy both global and local formulas.*

Proof. Suppose that, given a node $n_j \in N$, n_i is a child node of n_j satisfying $(n_j, n_i) \in E$, and they are in the path from n_0 to n_{min}. Due to lines 7 and 12 in Alg. 3, if $ap_i \in AP_G$, there exists $ap_i \in \Delta_G(s_j^{GT}, s_i^{GT})$, and if $ap_i \in AP_L$, there exists $ap_i \in \Delta_L(s_j^{LT}, s_i^{LT})$. Besides, if $ap_i \notin AP_G$, there exists $s_i^{GT} = s_j^{GT}$, and if $ap_i \notin AP_L$, there exists $s_i^{LT} = s_j^{LT}$. As n_j and n_i are in the path from n_0 to n_{min}, assume that n_i corresponds to the planning tuple $\Pi_k \in \Pi_{tem}$, and n_j corresponds to $\Pi_{k-1} \in \Pi_{tem}$. Therefore, $\forall s_k^L \in \boldsymbol{s}_L, k \in \mathbb{N}^+$, it holds that $\pi_k \in \Delta_L(s_{k-1}^L, s_k^L)$ or $\pi_k \notin AP_L$ and, $\forall s_k^G \in \boldsymbol{s}_G, k \in \mathbb{N}^+$, it holds that $\pi_k \in \Delta_G(s_{k-1}^G, s_k^G)$ or $\pi_k \notin AP_G$. Hence, the joint sampling is valid, i.e., $\Pi_{tem} \models (\Phi_G, \Phi_L)$. □

[1]For an LTL task ϕ and its corresponding NBA B_G, if there exist finite sequences $\boldsymbol{\pi} = \pi_2 \dots \pi_n$ and $\boldsymbol{s} = s_1 s_2 \dots s_{n-1} s_n$ that satisfy $\pi_i \in \Delta(s_{i-1}, s_i)$, $\forall \pi_i \in \boldsymbol{\pi}, s_i \in \boldsymbol{s}$, $s_n \in F_G$ and $s_i \in S_G$, it is said F_G is reachable from s_1.

Algorithm 3: Joint_Sampling

Input: Φ_G, Φ_L, AP
Output: Π_{tem}, Π_{pre}, Π_{tra}, Π_{suf}

1 Convert Φ_G, Φ_L to B_G, B_L;
2 Initialize $N = \{n_0\}$;
3 **for** $i = 1 : step_{max}$ **do**
4 $n_j = \mathsf{Sample}(N)$;
5 $ap_i = \mathsf{Sample}(AP)$;
6 **if** $ap_i \in AP_L$ **then**
7 $s_i^{LT} = \mathsf{Sample}(S_L)$, s.t. $ap_i \in \Delta(s_j^{LT}, s_i^{LT})$;
8 **else**
9 $s_i^{LT} = s_j^{LT}$;
10 **end**
11 **if** *if* $ap_i \in AP_G$ **then**
12 $s_i^{GT} = \mathsf{Sample}(S_G)$, s.t. $ap_i \in \Delta(s_j^{GT}, s_i^{GT})$;
13 **else**
14 $s_i^{GT} = s_j^{GT}$;
15 **end**
16 $n_i = (ap_i, s_i^{GT}, s_i^{LT})$;
17 Add (n_j, n_i) into E;
 $C(n_i) = C(n_j) + \|LM(ap_i) - LM(ap_j)\|$;
18 $\mathsf{Pruning}(\mathcal{T})$;
19 **end**
20 Get $n_{min} = (ap_{min}, s_{min}^{GT}, s_{min}^{LT}) \in N$, s.t. $s_{min}^{LT} \in F_L$, s_{min}^{GT} is reachable to F_G;
21 Obtain Π_{tem} from n_0 to n_{min} according to E;
22 Initial root node of sampling $N = \{n_0\} = (ap_{min}, s_{min}^{GT})$;
23 $[\Pi_{pre}, \Pi_{tra}, \Pi_{suf}] = \mathsf{Sampling}(\Phi_G)$;
24 Return Π_{tem}, Π_{pre}, Π_{tra}, Π_{suf};

Theorem 2. *The joint sampling in Alg. 3 is complete, i.e., if there exists a feasible planning scheme corresponding to the global formula and the local formula, the joint sampling is guaranteed to find it.*

Proof. Suppose $\Pi_{tem} = (\boldsymbol{s}_G, \boldsymbol{s}_L, \boldsymbol{p}, \boldsymbol{\pi})$ is a feasible local plan. Without considering the function Pruning and given a $\Pi_i = (s_i^G, s_i^L, p_i, \pi_i)$, if there exists a node $n_j = (ap_j, s_j^{GT}, s_j^{LT})$ such that $s_i^G = s_j^{GT}$ and $s_i^L = s_j^{LT}$, then according to lines 6-15 in Alg 3, there must exists a node $n_k = (ap_k, s_k^{GT}, s_k^{LT})$ that satisfies $s_{i+1}^G = s_k^{GT}$ and $s_{i+1}^L = s_k^{LT}$. If considering the function Pruning, the transitions that induce larger cost values will be removed. Therefore, if there does not exist a node n_k in the following sample from n_j, there must exists another node $n_l = (ap_l, s_l^{GT}, s_l^{LT})$, satisfying $s_{i+1}^G = s_l^{GT}$ and $s_{i+1}^L = s_l^{LT}$. As there must exists $s_0^G = s_0^{GT}$ and $s_0^L = s_0^{LT}$, for any $\Pi_i = (s_i^G, s_i^L, p_i, \pi_i) \in \Pi_{tem}$, there exists $n_l = (ap_l, s_l^{GT}, s_l^{LT})$ satisfying $s_{i+1}^G = s_l^{GT}$ and $s_{i+1}^L = s_l^{LT}$. Therefore, if there exists a feasible planning, there must exists a node $n_{min} = (ap_{min}, s_{min}^{GT}, s_{min}^{LT})$ such that $s_{min}^{LT} \in F_L$ and F_G is reachable from $s_{min}^{GT} = s_{|\Pi_{tem}|}^G$. Hence, the joint sampling is complete. □

According to the function Pruning, if there exist nodes n_i and n_j satisfying $ap_i = ap_j$, $s_i^{LT} = s_j^{LT}$, $s_i^{GT} = s_j^{GT}$, $C(n_i) \leq C(n_j)$, then n_j will be removed from N. Therefore, it must hold that $|N| \leq |S_G| \times |S_L| \times |AP|$, i.e., the maximum number of nodes in N is $|S_G| \times |S_L| \times |AP|$. Therefore, the complexity of joint sampling method is $O(n^3)$.

D. Real-Time Planning

Using the proposition graph and the joint sampling, the agent can realize proactive decision making for reactive local tasks via real-time planning. As indicated in Alg. 4, we first obtain the initial set of atomic proposition AP_G, the initial planning scheme Π, and the initial trajectory P (lines 1-3). First (line 5), the agent interacts with the environment based on local observation Ω. In lines 6-10, the agent updates the task plan Π based on Ω. The formula of local task Φ_L and the set of local propositions AP_L are inferred by Decision module, where the involved proposition AP_L of Φ_L can be obtained by function Get_ap. Finally, we obtain the new planning scheme $\Pi_{tem}, \Pi_{pre}, \Pi_{tra}, \Pi_{suf}$ by the plan module, which is then added into the global plan Π. In line 11-17, the trajectory P will be updated according to Π and the robot will navigate according to it.

Algorithm 4: Real-Time Planning

Input: P_G, Φ_G
Output: track P

1 $AP_G = \mathsf{Get\_ap}(\Phi_G)$;
2 $\Pi = \mathsf{Sampling}(\Phi_G, AP_G)$;
3 Initial P by Π;
4 **while** *1* **do**
5 $\quad \Omega = \mathsf{Get\_input}()$;
6 $\quad \Phi_{sensor}, \Phi_L = \mathsf{Decision}(PG, \Omega)$;
7 $\quad AP_L = \mathsf{Get\_ap}(\Phi_L)$;
8 $\quad AP = AP_G \cup AP_L$;
9 $\quad \Pi_{tem}, \Pi_{pre}, \Pi_{tra}, \Pi_{suf} =$ $\mathsf{Joint\_Sampling}(\Phi_G, \Phi_L, AP)$;
10 $\quad \Pi = \mathsf{Update}(\Pi, \Pi_{tem}, \Pi_{pre}, \Pi_{tra}, \Pi_{suf})$;
11 $\quad$ **for** i *in* $|\Pi|$ **do**
12 $\quad\quad$ **if** $i > 0$ **then**
13 $\quad\quad\quad P_{part} = \mathsf{RRT}(p_{i-1}, p_i)$;
14 $\quad\quad\quad$ Add P_{part} to P;
15 $\quad\quad$ **end**
16 $\quad$ **end**
17 $\quad \mathsf{Control}(P)$;
18 **end**

V. EXPERIMENT

LTL2STAR is used to convert LTL formula to Büchi automaton [26]. Python 3.8 and Matlab 2019b are used for experiment. The experiment video is provided[2].

As shown in Fig. 3(a), the environment has four areas of interest, which correspond to 4 atomic propositions. The global task is $\Phi_G = GFap_1 \wedge GFap_2 \wedge GFap_3 \wedge GFap_4$. For potential local tasks, we consider the following rules. If detecting a chair (i.e., ap_5), the robot should go to ap_1. If detecting a person (i.e. ap_6), the robot should go to ap_6, ap_2, and ap_1, where ap_1 has to be visited after ap_6 and ap_2 should be visited before ap_6. If detecting a backpack (i.e., ap_7), the robot should go to ap_3 and ap_7, where ap_3 should be visited after ap_7. Then, the PG can be defined as $re(ap_5) = \{ap_1\}$, $re(ap_6) = \{ap_1, ap_2, ap_6\}$, $re(ap_7) = \{ap_2, ap_7\}$, $af(ap_6) = \{ap_1\}$, $af(ap_7) = \{ap_3\}$, $bf(ap_6) = \{ap_2\}$.

[2] https://youtu.be/ffBURKa-008

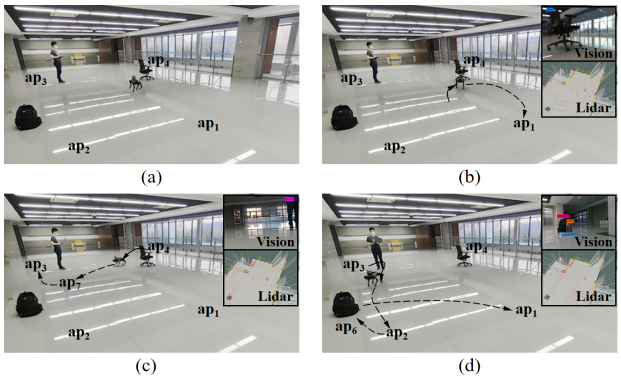


Fig. 3. The snapshots of experiments. In (a), the environment has 4 areas of interest, which correspond to 4 atomic propositions. In (b), the agent detects ap_5 and obtains the local task $\Phi_L = Fap_1$, then goes to ap_1. In (c), the agent detects ap_7 and obtains the local task $\Phi_L = Fap_7 \wedge Fap_3 \wedge ((\neg ap_3)Uap_7)$, then goes to ap_7 and ap_2. In (d), the agent detects ap_6 and obtains the local task $\Phi_L = Fap_6 \wedge Fap_2 \wedge Fap_1 \wedge ((\neg ap_6)Uap_2) \wedge ((\neg ap_1)Uap_6)$, then goes to ap_2, ap_6, ap_1.

The location of ap_5, ap_6, ap_7 are unknown a priori and can only be detected during runtime.

The experiment result are shown in Fig. 3 (b)-(d). To perform Φ_G, the global plan is generated as $ap_4ap_3ap_2ap_1$. During task operation, the agent may detect objects of interest. If the objects are in the PG, the local task will be triggered. For instance, as shown in Fig. 3(b), when going to ap_4, a chair is detected, which is related to the atomic proposition ap_5. Based on PG, the agent creates the local task $\Phi_L = Fap_1$ and the new task sequence is updated to $\Pi_{tem}\Pi_{pre} = ap_1ap_4ap_3ap_2ap_1$ and $\Pi_{tra} = \Pi_{suf} = ap_4ap_3ap_2ap_1$ by Alg. 3 within 0.242s. Then, the agent will go to ap_1 and go back to ap_4.

When going to ap_3, the robot then detects a person (i.e., ap_7), as shown in Fig. 3(c). Based on PG, the agent creates a local task $\Phi_L = Fap_7 \wedge Fap_3 \wedge ((\neg ap_3)Uap_7)$, which gives rise to the new task sequence $\Pi_{tem}\Pi_{pre} = ap_7ap_3ap_2ap_1$ and $\Pi_{tra} = \Pi_{suf} = ap_4ap_3ap_2ap_1$ by Alg. 3 within 0.265s. Then, the agent will go to ap_7 first and then ap_3. When going to ap_2, the backpack (i.e. ap_6) is detected, as shown in Fig. 3(d). Based on PG, the agent generates the local task $\Phi_L = Fap_6 \wedge Fap_1 \wedge Fap_2 \wedge ((\neg ap_6)Uap_2) \wedge ((\neg ap_1)Uap_6)$, and the new task sequence $\Pi_{tem}\Pi_{pre} = ap_2ap_6ap_1$ and $\Pi_{tra} = \Pi_{suf} = ap_4ap_3ap_2ap_1$ can be obtained by Alg. 3 within 0.304s. Then, the agent will go to ap_6, ap_3, and ap_1 sequentially. During the execution of global task, the agent detects three related objects and executes three local tasks in total, without violating the global task specification.

VI. CONCLUSIONS

This work develops a real-time decision-making and motion planning framework, which allows the robot to follow global task planned offline while autonomously generating an LTL formula for the local temporary task when encountering dynamic events. Future research will consider exploiting the proposition graph for more complex tasks and environments.

REFERENCES

[1] Z. Zhou, S. Wang, Z. Chen, M. Cai, H. Wang, Z. Li, and Z. Kan, "Local observation based reactive temporal logic planning of human-robot systems," *IEEE Trans. Autom. Sci. and Eng.*, 2023.

[2] Z. Zhou, Z. Chen, M. Cai, Z. Li, Z. Kan, and C.-Y. Su, "Vision-based reactive temporal logic motion planning for quadruped robots in unstructured dynamic environments," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 5983–5992, 2024.

[3] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1487–1507, 2021.

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

[5] 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.

[6] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for multi-agent systems from stl and spatel specifications," in *Proc. IEEE Conf. Decis. Control*. IEEE, 2017, pp. 4452–4457.

[7] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 48–61, 2009.

[8] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local ltl specifications," *Int. J. Robotics Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[9] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible ltl constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, 2023.

[10] H. Riener, "Exact synthesis of ltl properties from traces," in *2019 Forum for Specification and Design Languages (FDL)*. IEEE, 2019, pp. 1–6.

[11] D. Neider and I. Gavran, "Learning linear temporal properties," in *2018 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2018, pp. 1–10.

[12] C. Zheng and P. Kordjamshidi, "Relevant commonsense subgraphs for" what if..." procedural reasoning," *arXiv preprint arXiv:2203.11187*, 2022.

[13] S. Friedman, I. Magnusson, V. Sarathy, and S. Schmer-Galunder, "From unstructured text to causal knowledge graphs: A transformer-based approach," *arXiv preprint arXiv:2202.11768*, 2022.

[14] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "Atomic: An atlas of machine commonsense for if-then reasoning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 3027–3035.

[15] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments," *Int. J. Robotics Res.*, vol. 33, no. 12, pp. 1593–1607, 2014.

[16] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Robotics Res.*, vol. 39, no. 8, pp. 1002–1028, 2020.

[17] Z. Li, M. Cai, S. Xiao, and Z. Kan, "Online motion planning with soft metric interval temporal logic in unknown dynamic environment," *IEEE Control. Syst. Lett.*, vol. 6, pp. 2293–2298, 2022.

[18] H. Rahmani and J. M. O'Kane, "Optimal temporal logic planning with cascading soft constraints," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst.* IEEE, 2019, pp. 2524–2531.

[19] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *Proc. Int. Conf. Robot. Autom.* IEEE, 2021, pp. 12 618–12 624.

[20] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robotics Res.*, vol. 39, no. 7, pp. 812–836, 2020.

[21] V. Vasilopoulos, Y. Kantaros, G. J. Pappas, and D. E. Koditschek, "Reactive planning for mobile manipulation tasks in unexplored semantic environments," in *Proc. Int. Conf. Robot. Autom.* Xi'an, China: IEEE, 2021, pp. 6385–6392.

[22] B. Finkbeiner, F. Klein, and N. Metzger, "Live synthesis," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2021, pp. 153–169.

[23] G. F. Schuppe and J. Tumova, "Multi-agent strategy synthesis for ltl specifications through assumption composition," in *Proc. Int. Conf. Autom. Sci. Eng.* IEEE, 2020, pp. 533–540.

[24] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 2, pp. 494–514, 2021.

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

[26] P. Gastin and D. Oddoux, "Fast ltl to büchi automata translation," in *Int. Conf. Comput. Aided Verif.* Springer, 2001, pp. 53–65.

[27] Z. Chen, Z. Zhou, S. Wang, and Z. Kan, "A hierarchical decoupling approach for fast temporal logic motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2023, pp. 1579–1585.

[28] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe ltl specifications," in *IEEE Int. Conf. Intell. Robot. Syst.* IEEE, 2014, pp. 1511–1516.

[29] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2308–2315, 2017.

[30] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.