2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11247483

Reactive Temporal Logic Planning for Safe Human-Robot Interaction

Xiangcheng Liu, Ziyang Chen, Yinxiao Tian, and Zhen Kan

***Abstract*— Human-robot interaction plays a critical role in scientific experiments by ensuring efficient and reliable execution of experimental tasks. To achieve successful task completion, robots must adapt in real time to unexpected task variations, external disturbances, and safety constraints. In this work, we propose a reactive task and motion planning framework designed to address these challenges. By formulating interaction tasks using Linear Temporal Logic (LTL), our approach introduces Planning Decision Tree and Augmented Planning Decision Tree approach to dynamically adjust task sequences in response to environmental changes. At the execution layer, we employ a Model Predictive Path Integral controller, which ensures both efficient and safe control. Additionally, the planning interface effectively coordinates the planning and execution layers, ensuring strict adherence to experimental task specifications. The effectiveness of the proposed reactive planning framework is demonstrated through physical experiments using a 7-DoFs robot. Project website: https://sites.google.com/view/rtlp-iros/**

I. Introduction

In scientific experiments [1], [2], the integration of robotic systems as carriers, combined with human-robot interaction, has emerged as a novel research paradigm. A critical challenge in this domain is ensuring experimental safety and adherence to procedural standards while enabling robots to effectively manage the dynamic uncertainties inherent in human-robot interactions. Specifically, robots must accurately adjust task planning and control strategies to maintain flexibility in response to environmental variations. In human-robot interaction scenarios, robots must strictly adhere to temporal logic constraints to successfully execute designated experimental tasks. Any deviation from these constraints during task execution can lead to significant operational failures and severe consequences. Additionally, human operators may dynamically introduce temporary tasks based on experimental progress, requiring the robot to resolve potential conflicts between global and local temporary task objectives while ensuring adherence to safety constraints.

To deal with task changes, in [2], a time-varying cost function was designed to adjust task priorities in response to changes. However, this approach is suboptimal when handling randomly triggered tasks, as it necessitates manually predefined and complex state transition relationships. Furthermore, due to the rapid expansion of the symbol space, PDDL solvers cannot guarantee the system's real-time responsiveness. Alternative approaches, such as those in [3]–[5], leverage behavior trees, observation updates, and free-energy minimization to adapt to unforeseen local actions and resolve task conflicts. However, these methods depend on precise estimations of the robot's belief state, which is often impractical in real-world applications.

The rich expressiveness of Linear Temporal Logic (LTL) makes it a widely utilized framework for specifying complex robotic tasks [6]–[11]. LTL-based reactive planning algorithms dynamically adapt task execution in response to environmental and task variations, ensuring robust system performance across diverse conditions [12]–[22]. When responding to transient dynamic events [12], [15], the robot must accurately assess the current task progress to ensure continuous task advancement [18], [19], [23]. Sampling-based methods require searching for solutions over the product of Büchi automata and the transition system [7], which imposes significant computational overhead. Prior works [12], [13] have demonstrated the capability to manage localized single tasks in dynamic environments; however, they struggle with scenarios where local tasks are interdependent with global task objectives. A reconfigurable planning approach based on behavior trees [15] enables robots to adapt and plan with minimal replanning steps by leveraging past experiences. However, this method relies on the cross-product of states and automata, which significantly constrains search efficiency when handling complex local task specifications. Reactive synthesis and probabilistic synthesis allow robots to adjust their actions flexibly to satisfy given task specifications, optimizing both planning efficiency and strategy effectiveness [20]–[22]. However, these approaches are generally limited to finite time horizons, whereas real-world robotic applications often involve tasks with infinite time horizons. In human-robot interaction scenarios, where robots encounter dynamic disturbances, sampling-based model predictive control approaches, such as Model Predictive Path Integral (MPPI) control [24], effectively handle nonlinearities and nonconvexities by sampling control inputs and simulating system dynamics to approximate optimal control strategies [25], [26] that ensure both interaction safety.

Motivated by the discussion above, this work aims to develop a reactive task and motion planning framework for experimental scenario that can enable plan adjustments, ensure human operator safety and accommodate unexpected task variations in real time. The contributions are summarized as follows. First, a temporal logic-based planning approach, incorporating a planning decision tree and an augmented planning decision tree search method, is developed to dynamically adjust task sequences in response to changes in local temporary tasks and task-related objects during human-robot

This work was supported in part by National Key R&D Program of China under Grant 2022YFB4701400/4701403.

X. Liu, Z. Chen, Y. Tian, and Z. Kan are with the Department of Automation at the University of Science and Technology of China, Hefei, Anhui, China, 230026.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

interaction. This approach is particularly effective in handling scenarios where global and local tasks are interdependent. Second, a hierarchical framework is proposed to coordinate the task planning and execution layers, ensuring both the safety of human-robot interaction and strict adherence to task specifications. Third, the effectiveness of the proposed method is validated through physical experiments.

II. PRELIMINARIES

LTL is a formal language capable of expressing rich task specifications. Given the atomic propositions AP, the syntax of the LTL formula is defined as

$$\phi := \textit{true} \mid ap \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid X\phi \mid \phi_1 \mathcal{U} \phi_2,$$

where *true* is the Boolean value, $ap \in AP$ is an atomic proposition, ϕ is an LTL formula, $\neg$ (negation) and $\wedge$ (conjunction) are standard Boolean operators, and X (next) and U (until) are temporal operators. Any LTL formula can be converted to a Nondeterministic Büchi Automata (NBA).

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

The word $\boldsymbol{\pi} = \pi_0\pi_1\ldots$ 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. Let $\Delta : S \times S \rightarrow 2^{\Sigma}$ denote the set of atomic propositions that enables state transitions in NBA, i.e., $\forall \pi \in \Delta(s, s'), s \xrightarrow{\pi}_B s'$. Then, the NBA can also be defined as $B = (S, S_0, \Sigma, \Delta, S_F)$. A run $s = s_0 s_1 s_2 \ldots$ of B generated by the word $\boldsymbol{\pi}$ is called valid, if $s_{i-1} \xrightarrow{\pi_i} s_i, \forall i \in \mathbb{N}_{\geq 1}$, and s intersects with S_F infinitely many times. Co-safe LTL is a special case of LTL, where its satisfaction is defined by a finite sequence of word. More details on LTL syntax, semantics, and model checking can be found in [27]. In this work, we consider a global LTL task Φ_G and a temporary local task Φ_L, whose corresponding NBA is $B_G = (S_G, S_{G0}, \Sigma_G, \Delta_G, F_G)$ and $B_L = (S_L, S_{L0}, \Sigma_L, \Delta_L, F_L)$, respectively. Since Co-safe LTL is suitable to describe finite length robotic tasks, the local temporary tasks Φ_L are specified by co-safe LTL formulas. Let $\mathbb{Z}_{\geq 0}$, $\mathbb{N}$, $\mathbb{R}^+$, and $[N]$ denote the set of non-negative integers, the set of natural numbers, the set of positive real numbers, and the shorthand notation for $\{1, \ldots, N\}$, respectively. Given a set A, denote by $|A|$ the cardinality of A.

III. PROBLEM FORMULATION

The state space of the robot, denoted as $P \subset \mathbb{R}^7$, represents the pose state of the robot's end-effector. Specifically, a state $p \in P$ is defined as $p = [x, y, z, q_w, q_x, q_y, q_z]^\top$. where x, y, z are the Cartesian coordinates of the position, and q_w, q_x, q_y, q_z are the components of the unit quaternion representing the orientation. Consider a n DoFs robot manipulator with K rigid links operating in a bounded workspace $M \subset \mathbb{R}^3$ containing $\{m_i\}_{i=1,\ldots,n}$ regions of interest (e.g., cuboid regions) associated with the robot state space P. We denote by $L_{pm} : P \rightarrow M$ a label function, which establishes a one-to-one correspondence between an robot's state $p_i \in P$ and a region of interest m_i. Let O be the set of observed objects, where $o_i \in O$ represents the ith object. Let $L_{mo} : M \times O \rightarrow \{0, 1\}$ be a Boolean indicator function that indicates whether an object o_i is in a region m_i. The robot configuration is defined as $x \in \mathbb{R}^n$. Each link has a rigidly attached coordinate system $\{S_k\}_{k=1}^K$, with S_0 as the base frame. The homogeneous transformation matrix $\mathbf{T}_k \in SE(3)$, mapping the base frame S_0 to the frame S_k of the k-th link, is computed using Denavit-Hartenberg (DH) parameters and joint configuration x as part of the forward kinematics : $\mathbf{T}_k = \text{FK}_k(x)$. To facilitate robot tasks and motion planning, the following transition system is defined.

Definition 2. The transition system (TS) is a tuple $\mathcal{T} = (P, p_0, \rightarrow_{\mathcal{T}}, AP, L, L_M, C_{\mathcal{TS}})$, where P is a set of pose states; $p_0 \in P$ is the initial state; $\rightarrow_{\mathcal{T}} \subseteq P \times P$ indicates the state transition such that $(p, p') \in \rightarrow_{\mathcal{T}}$ if the transition is feasible; AP is the set of atomic propositions (i.e. reaching a specified location or completing a desired task); $L : P \rightarrow AP$ is a labeling function that indicates the atomic proposition $ap \in AP$ at the state p, $L_M : AP \rightarrow M$ indicates the interested region associated with the atomic proposition; and $C_{\mathcal{TS}} : (\rightarrow_{\mathcal{T}}) \rightarrow \mathbb{R}^+$ is the cost of transition, (i.e. the Cartesian distance between states p and p'). Assuming that the path between p and p' consists of discrete points $\gamma_i \in \mathbb{R}^3, i = 1, \ldots, n$, corresponding to the robot's Cartesian positions, the transition cost is defined as $C_{\mathcal{TS}}(p, p') = \sum_{k=1}^{n-1} \|\gamma_{k+1} - \gamma_k\|$, $\forall (p, p') \in \rightarrow_{\mathcal{T}}$.

The plan of global task Π is defined as $\Pi = (\boldsymbol{s_G}, \boldsymbol{p}, \boldsymbol{\pi})$, where $\boldsymbol{s_G} = s_0^G s_1^G s_2^G \ldots$ is the sequence of states of B_G, $\boldsymbol{p}$ represents the sequence of states, and $\boldsymbol{\pi}$ is the word defined before. By [27], the plan Π can be written in the form of prefix-suffix structure $\Pi = \Pi_{pre}[\Pi_{suf}]^w$, where Π_{pre} is the prefix part starting from an initial state and ending at an accepting state and Π_{suf} is the suffix part with the same starting and ending NBA state. There exists $\pi_i \in \Delta_G(s_{i-1}^G, s_i^G)$, $\forall i \in [|\Pi|]$, and $s_{|\Pi_{pre}|}^G = s_{|\Pi_{suf}|}^G \in F_G$. The cost of plan $\boldsymbol{p}$ is defined as $J(\boldsymbol{p}) = \sum \|p - p'\|$, $\forall (p, p') \in \rightarrow_{\mathcal{T}}$. Consequently, the cost of a prefix-suffix plan Π can be defined as $J(\boldsymbol{p}) = wJ(\boldsymbol{p}^{\text{pre}}) + (1 - w)J(\boldsymbol{p}^{\text{suf}})$ [7], where w is the tuning weight. Given a local task Φ_L, the local plan is defined as $\Pi_{tem} = (\boldsymbol{s_G}, \boldsymbol{s_L}, \boldsymbol{p}, \boldsymbol{\pi})$, where $\boldsymbol{s_L} = s_0^L s_1^L s_2^L \ldots$ is the sequence of states of B_L. When a temporary task Φ_L is activated, Π should satisfy Φ_L while respecting Φ_G. A temporal plan satisfies both Φ_G and Φ_L if, $\forall i \in [n]$, $\pi_i \in \Delta_L(s_{i-1}^L, s_i^L)$, $\pi_i \in \Delta_G(s_{i-1}^G, s_i^G)$ and $s_n^L \in F_L$, which is denoted as $\Pi_{tem} \models (\Phi_G, \Phi_L)$.

Assumption 1. There exists a plan Π satisfying task Φ_G and Φ_L during human-robot interactions.

Problem 1. Given global task Φ_G and local temporary task Φ_L, the goal is to design a real-time reactive framework that can determine and execute an infinite sequence $\boldsymbol{x}$ that minimizes $J(\boldsymbol{p})$ while ensuring: (1) the completion of Φ_G; (2) the completion of local temporary task Φ_L while not violating Φ_G; (3) avoiding collision and adjusting plan Π

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

during human-robot interaction .

IV. Reactive Planning for Safe Human-Robot Interaction

This section introduces a real-time reactive planning framework designed to accommodate environmental and task variations in human-robot interactions while ensuring an optimal and safe path $\boldsymbol{p}$. As illustrated in Fig. 1, the *task planner* manages both the global task Φ_G and the local task Φ_L, transforming them into their corresponding NBA representations B_G and B_L. An initial offline and optimal task plan Π is then generated using a planning decision tree search. To handle temporary tasks and dynamic environmental changes, we further develop an augmented planning decision tree along with *motion generator* to enable rapid real-time adaptation of $\boldsymbol{p}$ while ensuring collision-free motion. The *planning interface* coordinates the planning and execution layer, ensuring adherence to task constraints.

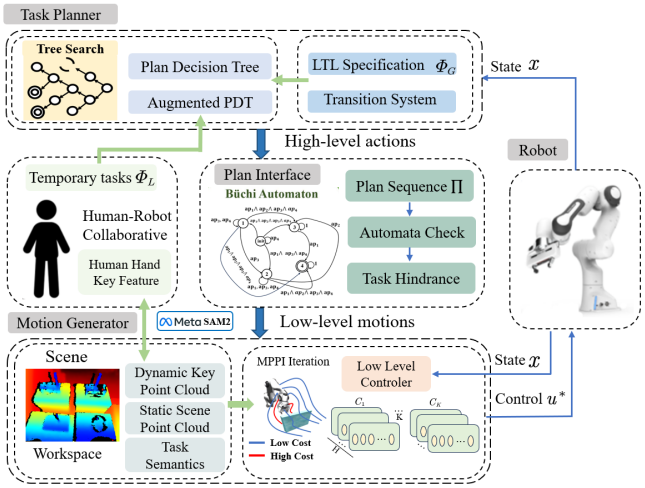

Fig. 1: The framework of real-time reactive planning.

A. Task Planner

To obtain an optimal task plan Π that satisfies Φ_G in real time, inspired by previous work [11], we construct a planning decision tree to represent the task progress and expand nodes that align with automaton transitions. This approach restricts the search to relevant automaton states and sub-tasks, thereby avoiding constructing and traversing the product automaton.

Definition 3. The planning decision tree $\mathcal{T}_P$ is constructed based on a set of nodes $\{d_i^P\}$, $i \in \mathbb{Z}_{\geq 0}$, where d_0^P represents the root, and each node in $\mathcal{T}_P$ is defined as a tuple

$$d_i^P = (s_i, p_i, ap_i, \mathrm{Ori}(d_i^P), \mathrm{Tra}(d_i^P), \mathrm{C}(d_i^P), \mathrm{Flag}(d_i^P))$$

where $s_i \in S$ denotes the automaton state, $p_i \in P$ denotes the robot pose state, $ap_i \in AP$ denotes the atomic proposition, $\mathrm{Ori}(d_i^P)$ denotes the parent node of d_i^P, $\mathrm{Tra}(d_i^P) \in \{0, 1\}$ indicates whether d_i^P can generate its child nodes and $\mathrm{Tra}(d_i^P) = 0$ by default, $\mathrm{C}(d_i^P)$ represents the total Cartesian distance. Specifically, for a node sequence $d_{i_0}^P d_{i_1}^P \dots d_{i_n}^P$, where $i_0 = 0$, $i_n = i$, the cost is defined as $\mathrm{C}(d_i^P) = \Sigma_{j=1}^{n} C_{\mathcal{TS}}(p_{i_j}, p_{i_{j-1}})$. It is simulated and computed through sampling method based on GPU acceleration [28]. For the initial static environment, $C_{\mathcal{TS}}(p_{i_j}, p_{i_{j-1}})$ will be saved through cache to avoid repeated calculation overhead.

Algorithm 1 Planning Decision Tree

Input: Global Task Φ_G, Workspace M, Initial Node d_0
Output: Π_{pre}, Π_{suf}

Convert Φ_G to NBA B_G;
Initialize the tree $\mathcal{T}_P = \{d_0^P\}$
while True **do**
  if $\mathrm{Tra}(d^P) = 1,\ \forall d_i^P \in \mathcal{T}_P$ **then**
    break
  # Generate Children Node By Tree Expansion
  for $d_i^P \in \mathcal{T}_P$ **s.t.** $\mathrm{Tra}(d_i^P) = 0$ **do**
    for $s \in S \setminus \mathrm{PreS}(d_i^P)$ **do**
      for p s.t. $L(p) \in \Delta(s_i, s) \wedge L_{mo}(L_{pm}(p), o_i)$ **do**
        Add d_{chi}^P to $\{d_{\text{chi}}^P\}$ and Update $\mathrm{PreS}(d_{\text{chi}}^P)$
  # Avoid Unnecessary Expansion By Tree Pruning
  for d_{sub}^P in $\{d_i^{\text{chi}}\}$ **do**
    if $\mathrm{Prog}(d_{\text{sub}}^P) = \mathrm{oth}$ **then** $\mathrm{Tra}(d_{\text{sub}}^P) = 1$
    for d_i^P in $\mathcal{T}_P$ **do**
      if $s_i = s_{\text{sub}}$ and $\mathrm{Prog}(d_i^P) = \mathrm{Prog}(d_{\text{sub}}^P)$ **then**
        if $\mathrm{C}(d_i^P) < \mathrm{C}(d_{\text{sub}}^P)$ **then**
          $\mathrm{Tra}(d_{\text{sub}}^P) = 1$;
          Delete d_{sub}^P and all its leaf nodes in $\mathcal{T}_P$
  Add d_i^{sub} to $\mathcal{T}_P$
  Set $\mathrm{Tra}(d_i^P) = 1$
$\Pi_{pre}, \Pi_{suf} = \mathrm{Get\_Plan}(\mathcal{T}_P)$
Return Π_{pre}, Π_{suf}

Let $\mathrm{Prog}(d_i^P) \in \{pre, oth\} \cup S_F$ indicate the associated task stage, where pre and oth represent the prefix stage and the completion of first suffix stage, respectively, and $s \in S_F$ indicates that the plan is in the suffix stage with starting state s. The task stage $\mathrm{Prog}(d_i^P)$ evolves according to

$$\mathrm{Flag}(\mathrm{Prog}(d_i^P), s) = \begin{cases} \mathrm{Prog}(d_i^P), & \text{if } s \notin S_F, \\ s, & \text{if } s \in S_F, \\ \mathrm{oth}, & \text{if } \mathrm{Prog}(d_i^P) = s. \end{cases}$$

Let d_{chi}^P represent the child node generated by parent node d_{par}^P and $\mathrm{PreS}(d_{chi}^P)$ records the automaton states of nodes that share the same mission stage within the d_{par}^P. The $\mathrm{PreS}(d_{chi}^P)$ is updated based on $\mathrm{Prog}(d_{par}^P)$. If $\mathrm{Prog}(d_{chi}^P)$ changes, we reset $\mathrm{PreS}(d_{chi}^P) = \emptyset$. Otherwise, $\mathrm{PreS}(d_{chi}^P) = \mathrm{PreS}(d_{par}^P) \cup \{s_{chi}\}$, which avoids the same NBA states appearing in one task stage.

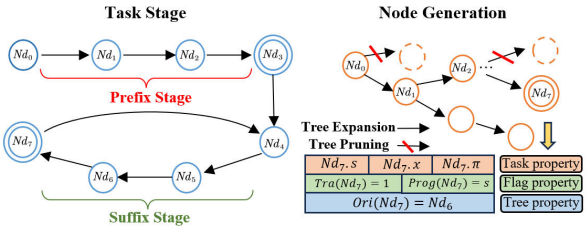

Fig. 2: The task stages of φ and the construction of planning decision tree by Tree Expansion and Tree Pruning.

The planning process is outlined in Alg. 1. As shown in Fig. 2, the tree $\mathcal{T}_P$ represents the task stages and system states

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

and generates children nodes incrementally from the initial node d_0^P including the initial state s_0, the initial position p_0, the initial proposition ap_0 with $\text{Prog}(d_0^P) = pre$ and $\text{PreS}(d_0^P) = \{s_0\}$. If d_{chi}^P has the same s and $Prog$ with a node d_i^P in $\mathcal{T}_P$ and $\text{C}(d_i^P) < \text{C}(d_{chi}^P)$, the d_{chi}^P will no longer be traversed, and the paths from the d_i^P to its leaf nodes will all be pruned out from $\mathcal{T}_P$. Finally, the function $\text{Get\_Plan}(\mathcal{T}_P)$ performs backtracking to the root node to obtain the plan sequence $\Pi \models \Phi_G$. In offline planning, $L_{mo}(L_{pm}(p), o_i)$ is set according to the initial state of objects o_i in the workspace M.

B. Reactive Task Plan

During task execution, the robot must satisfy the global task specification Φ_G while also addressing local temporary tasks Φ_L. This planning process must coordinate the two task requirements and ensure that no task constraints are violated in real-time. We extend the original planning decision tree $\mathcal{T}_P$ and consider the state transitions of the automata corresponding to the two tasks at the same time. We construct an augmented planning decision tree $\mathcal{T}_J$ comprising a set of nodes $\{d_i^J\}$, $i \in \mathbb{Z}_{\geq 0}$ with d_0^J being the root. Each node d_i^J is characterized by a tuple $d_i^J = (p_i, ap_i, s_i^G, s_i^L, C(d_i^J))$, where $ap_i \in AP$ and $p_i \in P$ are the same to $\mathcal{T}_P$, $s_i^G \in S_G$ denotes the global task state, $s_i^L \in S_L$ indicates the local task state. Similar to $\text{C}(d_i^P)$, $\text{C}(d_i^J)$ is the cost defined on a node sequence. For the node sequence $d_{i_0}^J d_{i_1}^J \dots d_{i_n}^J$, where $i_0 = 0$ and $i_n = i$, the cost is defined as $\text{C}(d_i^J) = \Sigma_{j=1}^n C_{\mathcal{TS}}(p_{i_j}, p_{i_{j-1}})$. A transition $(d_i^J, d_j^J) \in E$ is considered feasible if it satisfies both the local task Φ_L and the global task Φ_G. This is achieved when $ap_j \in \Delta_L(s_i^L, s_j^L)$ or $ap_j \notin AP_L$, and $ap_j \in \Delta_G(s_i^G, s_j^G)$ or $ap_j \notin AP_G$.

Algorithm 2 Augmented Planning Decision Tree

Input: Global Task Φ_G, Local Task Φ_L, AP
Output: $\Pi_{tem}, \Pi_{pre}, \Pi_{suf}$

Convert Φ_G, Φ_L to NBA B_L, B_G;
Initialize the tree $\mathcal{T}_J = \{d_0^J\}$
for $i = 1$ **to** $step_{max}$ **do**
  $d_j^J = \text{Sample}(\{d^J\})$, $s_i^L = s_j^L$, $s_i^G = s_j^G$
  $ap_i = \text{Sample}(AP)$
  if $ap_i \in \Delta_L(s_j^L, s_i^L)$ **then** $s_i^L = \text{Sample}(S_i^L)$;
  if $ap_i \in \Delta_G(s_j^G, s_i^G)$ **then** $s_i^G = \text{Sample}(S_i^G)$;
  if $L_{mo}(L_{pm}(p_i), o_i)$ **then**
    Add (d_j^J, d_i^J) **into** $Edge$ E;
    $\text{C}(d_i^J) = \text{C}(d_j^J) + C_{\mathcal{TS}}(d_i^J.p_i, d_j^J.p_j)$;
  # Pruning to avoid invalid states search
  for (d_j^J, d_i^J) in E **and** (d_j^J, d_i^J) is feasible **do**
    if $d_i^J \equiv d_j^J$ **and** $\text{C}(d_i^J) \leq \text{C}(d_j^J)$ **then**
      Delete d_j^J from $\{d^J\}$
    for d_k^J in $\{d_i^J\}$
    if $\text{C}(d_i^J) - \text{C}(d_k^J) > C_{\mathcal{TS}}(d_i^J.p_i, d_k^J.p_k)$ **then**
      Delete (d_j^J, d_i^J) from E, Add (d_k^J, d_i^J) to E
      $\text{C}(d_i^J) = \text{C}(d_k^J)) + C_{\mathcal{TS}}(d_i^J.p_i, d_k^J.p_k)$
Get d_{min}^J, s.t. $s_{\text{min}}^L \in F_L$ and s_{min}^G is reachable to F_G;
Obtain Π_{tem} **from** d_0^J **to** d_{min}^J according to E;
Initial root node $N = \{d_0^J\} = (ap_{min}, s_{min}^G)$;
$[\Pi_{\text{pre}}, \Pi_{\text{suf}}] = \text{Planning Decision Tree}(\Phi_G, \{d_{min}^J\})$; ▷ Algorithm 1
return $\Pi_{\text{tem}}, \Pi_{\text{pre}}, \Pi_{\text{suf}}$;

The augmented planning decision tree search based on $\mathcal{T}_J$ is outlined in Alg. 2, which achieves efficient local task planning by sampling atomic propositions AP and iterating through the states of automata B_G and B_L. This approach prunes nodes during the tree expansion process, avoiding the expansion of invalid states. After constructing the tree $\mathcal{T}_J$, the node d_{min}^J with the minimum $\text{C}(d_{min}^J)$ and also satisfy minimizes $J(\boldsymbol{p})$ is selected among those completing the local task Φ_L and global task Φ_L. For the corresponding state s_{min}^G, there exists a finite sequence of transitions $\mathbf{\Pi}$ and states $s = s_{min}^G, \dots, s_n^G$, where i is the index of the states , such that $\pi_i \in \Delta(s_{i-1}, s_i)$ for all i, and $s_n^G \in F_G$, ensuring consistency between local and global task completion.

C. Motion Generator

To enable reactive motion planning during human-robot interaction, accurately estimating the distance between robot, workspace and human operator is crucial. Specifically, the robot's workspace is represented as a dense set of points $\{\mathbf{o}_m \in \mathbb{R}^3\}_{m=1,\dots,O}$. Given a set of robot configurations $\{x_i\} \in X$, following the works of [25], a batch of N control points $\{\mathbf{k}_j(x) \in \mathbb{R}^3\}_{j=1,\dots,N}$ can be obtained for each configuration x by forward kinematics. The Configuration Signed Distance Function (CSDF) [29] is then employed,

$$\text{CSDF}(x) := \min_{j=1,\dots,N} \left(\min_{m=1,\dots,O} \|\mathbf{k}_j(x) - \mathbf{o}_m\| \right) - r, \quad (1)$$

where r represents a safety threshold. Positive CSDF indicates that the robot is safe and away from obstacles.

Model Predictive Path Integral (MPPI) is based on a discrete-time, continuous-state system: $x_{t+1} = x_t + e_t$ with $e_t \sim \mathcal{N}(d_t, \Sigma)$, where $\mathcal{N}(d_t, \Sigma)$ denotes a multivariate Gaussian distribution with mean d_t representing the nominal displacement from x_t at time t and covariance matrix Σ. We begin the MPPI loop by assuming a straight-line path between the initial and target configurations. The path is discretized into intermediate waypoints, and the corresponding nominal displacements are calculated to serve as the initial conditions for MPPI updates. At each MPPI iteration, Z candidate sequences of displacements $E_i = \{e_{i,h}\}_{h=0}^{H-1}$, $i = 1, \dots, Z$, are sampled from $\mathcal{N}(d_t, \Sigma)$ based on a series of nominal configurations trajectory $x_0, \dots, x_H$, and corresponding nominal displacements $d_0, \dots, d_{H-1}$ over a control horizon H. The sampling and evaluation processes are using GPU. The rollout cost $C(E_i)$ is the weighted sum of C_{terminal} and C_{run}. Specifically, the terminal cost

$$C_{\text{terminal}}(E_i) := w_{\text{terminal}} \|x_{i,H} - x_{\text{goal}}\| \quad (2)$$

quantifies the distance between the actual configuration at the end of the rollout and the goal configuration, where w_{terminal} is the tuning weight. For the running cost C_{run}, we consider weighted contributions from the total path length, collisions with the external environment, self-collisions, and the avoidance of stagnation. The running cost is defined as

$$C_{\text{run}}(E_i) = C^{\text{length}}(E_i) + C^{\text{coll}}(E_i) + C^{\text{self-coll}}(E_i) + C^{\text{stay}}(E_i) \quad (3)$$

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

$$C^{\text{length}}(E_i) := w_{\text{length}} \sum_{t=0}^{H-1} \|\text{FK}_K(x_{i,t} + e_{i,t}) - \text{FK}_K(x_{i,t})\|, \tag{4}$$

$$C^{\text{coll}}(E_i) := w_{\text{coll}} \sum_{t=0}^{H-1} c_{\text{coll}}(x_{i,t}, e_{i,t}), \tag{5}$$

$$C^{\text{self-coll}}(E_i) := w_{\text{self-coll}} \sum_{t=0}^{H-1} c_{\text{self-coll}}(x_{i,t}, e_{i,t}), \tag{6}$$

$$C^{\text{stay}}(E_i) := w_{\text{stay}} \frac{C^{\text{terminal}}(E_i)}{\max\left(\epsilon, \|x_{i,1} - x_{i,H}\|_2\right)}, \tag{7}$$

where w_{length}, w_{coll}, $w_{\text{self-coll}}$, w_{stay} are the tuning weights. $C^{\text{length}}(E_i)$ is to penalize long movement distance of the end effector in Cartesian space. In (5), c_{coll} is designed as

$$c_{\text{coll}}(x_{i,t}, e_{i,t}) := \begin{cases} 1 & \text{if CSDF}(x_{i,t}) \leq \delta, \\ \frac{\delta}{\text{CSDF}(x_{i,t})} & \text{if CSDF}(x_{i,t}) > \delta, \end{cases} \tag{8}$$

where the cost is set to 1 if the CSDF is below the threshold δ and decreases as CSDF rises. In (6), $c_{\text{self-coll}}$ is estimated through neural network as in [26]. In (7), C^{stay} penalizes static trajectories, encouraging exploration, reducing being stuck far from the goal. When trajectory converges to x_{goal}, this cost becomes zero. The cost of trajectory E_i is defined as $S(E_i) = C(E_i) + \lambda \sum_{t=0}^{H-1} d_t^\top \Sigma^{-1} e_{j,t}$. The exponential averaging $\Omega(E_i)$ is defined as

$$\Omega(E_i) := \exp\left(-\frac{1}{\lambda}\left(S(E_i) - \min_{i=1,\ldots,Z} S(E_i)\right)\right), \tag{9}$$

where λ is the temperature parameter influencing its exploration performance. The nominal displacements are then updated based on the exponential averaging of the rollout costs and a weighted sum of $\Omega(E_i)$. Specifically, at the ith iteration, the nominal distance d_t is updated according to

$$d_t := (1 - \alpha_d) d_{t-1} + \alpha_d \cdot \frac{\sum_{i=1}^{Z} \Omega(E_i).e_{i,t}}{\sum_{i=1}^{Z} \Omega(E_i)}, \tag{10}$$

where $\alpha_d \in (0, 1)$ is a tunable weight influencing the smoothness of the control input generation.

Given a set of discrete configurations trajectory $\boldsymbol{x}$ from the current configuration x to x_{goal}, x_{s^*} is identified as the farthest configuration from x that is still within a safe distance. Specifically, x_{s^*} is selected as

$$x_{s^*} := \arg\max_{x_i} \left\{\|x - x_i\| \mid \|\mathbf{k}(x_i) - \mathbf{k}(x)\| \leq \text{CSDF}(x)\right\}. \tag{11}$$

Once the goal configuration x_{s^*} is determined, a potential field for trajectory following is constructed following [25] as

$$\varphi(x) := \frac{\|x - x_{s^*}\|^2 + \epsilon}{\text{CSDF}(x) + \epsilon}, \tag{12}$$

where ϵ denotes a small constant that ensures numerical stability. Finally, the control input is designed along the gradient of the potential field as

$$u(x) := -k \nabla \varphi(x), \tag{13}$$

where k is a positive weight. By employing rolling optimization to determine the control input u and applying it to the robot, safe motion can be achieved. In human-robot interaction, the influence of the human is modeled as a dynamic point cloud which generated by dynamically tracking the human hand's mask [30] and integrated into the workspace point cloud to generate control sequence, ensuring safety during the human-robot interaction process.

The planning interface restricts the planning domain according to the high-level task sequence, ensuring task constraints are satisfied. In the process of motion generation, atomic proposition ap corresponding Cartesian regions $L_M(ap)$ that are restricted during the task transition must be treated as static obstacles and incorporated into the low-level controller. To this end, the hindrance in Π are defined as $HIN = hin_0 hin_1, \ldots$, where $hin_i \in 2^{AP}$ for $i = 0, 1, \ldots$, indicating the banned actions and associated areas during the transition from Π_i to Π_{i+1}. Specifically, the hindrance set HIN is constructed by identifying constraints during state transitions in the plan Π. For each state $\Pi_i = (s_i, x_i, \pi_i)$, the prohibited action $ap \in AP$ during the transition from s_{i-1} to s_i will be checked by Alg. 3. If $\pi_i \wedge ap$ is not in the transition set $\Delta(s_{i-1}, s_i)$, such actions are added to hin_i to ensure no prohibited actions occur. During the execution of $\Pi_i \to \Pi_{i+1}$, the transitions from p_i to p_{i+1} with $L_M(hin_i)$ as a static obstacle in MPPI planning ensures that the task constraints are met. When human-robot interaction exceeds the threshold time or causes changes in $L_{mo}(m_i, o_i)$, new control sequence $\boldsymbol{u}$ will be generated by combining algorithms 1, 2, and 3 to satisfy the minimization of $J(\boldsymbol{p})$.

Algorithm 3 Low-level Motion planning

Input: Plan Π, Büchi Automata B
Output: Hindrance Set HIN, Set of Control Inputs u

```
Initialize HIN ← ∅;
for each state Π_i = (s_i, p_i, π_i) do
    if ap is prohibited in transition s_{i-1} → s_i then
        if π_i ∧ ap ∉ Δ(s_{i-1}, s_i) then
            Add ap to hindrance hin_i;
    Compile hin_i into HIN;
for each plan Π_i → Π_{i+1} do
    if L(p_i) in HIN then
        Add L_M(hin_i) and Remove L_M(ap_i)
    else
        Add L_M(hin_i) as static obstacle;
    Get minimum norm solution x_{i+1} from (x_i, p_i);
    Generate the control input u by motion generator
return HIN, u;
```

V. EXPERIMENT

Consider a human-robot interaction scenario in chemical experimentation. The robot is responsible for sequentially performing tasks such as sampling, dispensing, and stirring within designated areas according to the experimental protocol. Additionally, the robot must handle unexpected sampling tasks while ensuring that these actions do not conflict with the overall experimental protocol. Compared with the related methods in Table I, our method can handle local tasks with complex and long-sequence task specifications. When

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

Method	Dynamic Scenario	Online Coordination			Product Automaton Free[1]	Inheritance Task Progress[2]
		Local Long Task	Local Single Task	Coupled Local Task		
Vasile [12]	✓	✗	✓	✗	✗	✓
Kantaros [13]	✓	✗	✓	✗	✓	✓
Finkbeiner [14]	✗	✓	✓	✗	✗	✓
Li [15]	✗	✗	✓	✗	✗	✓
Sahin [16]	✗	✗	✗	✗	✓	✗
Kurtz [17]	✗	✗	✗	✗	✓	✗
Ours	✓	✓	✓	✓	✓	✓

TABLE I: Comparison with Related Literature.

[1] Product Automaton Free means that there is no need for explicit product of transition system and automaton, which can reduce the search space.
[2] Inheritance Task Progress means that the current task progress can be preserved, and there is no need to start the search from scratch.

local tasks are coupled with global tasks, our method can coordinate the two types of tasks and ensure that both task specifications are strictly followed.

The global tasks Φ_G are represented by atomic propositions ap_i, $i = 1, \ldots, 6$, which correspond to the actions: taking a sample tube in Area 1 or Area 2, dispensing the sample in Area 3, placing the test tube in Area 1 or Area 2, picking up the stirring rod in Area 4, using the stirring rod to stir in Area 3, and placing the stirring rod back in Area 4, respectively. Initially, Alg. 1 is used to generate the offline task planning:

$$\Phi_G = G\Big(\Big(\bigwedge_{i=1}^{5}(ap_i \rightarrow Xap_{i+1})\Big) \wedge (ap_6 \rightarrow Fap_1)\Big) \wedge Fap_1$$

The local (temporary) tasks are specified via co-safe LTL as

$$\Phi_L = \bigwedge_{i=7,8,9} (\neg(ap_{i+1})U(ap_i \wedge \neg ap_{i+1})) \wedge Fap_{10},$$

where ap_7 and ap_9 represent the task of placing pill A or B in Area 3, and ap_8 and ap_{10} represent picking up pill A from Area 6 and pill B from Area 5, respectively. During the process of chemical experimentation, Φ_L can be triggered by the operator anytime based on the experiment's progress.

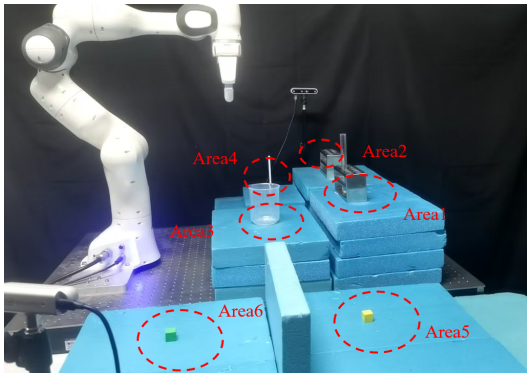

Fig. 3: The experimental setup, where each region corresponds to a different operation task.

In the experiment, we utilize a Franka Panda with NVIDIA RTX-3060. RealSense D455 and D435 RGB-D cameras are in an eye-to-hand setup to track the operator's hand point cloud [30] and monitor the spatial distribution of environmental objects [31]. Spacemouse is used as a trigger for local tasks. LTL2BA [32] is used to convert LTL formulas into Büchi automata. To perform Φ_G, by Alg. 1, $\Pi_{pre} = ap_0ap_1$ and $\Pi_{suf} = ap_2ap_3ap_4ap_5ap_6ap_1$ are generated offline within 0.14s. The local task Φ_L is then triggered by the operator in Fig. 4. For all p_i, $i \in \{1, \ldots, j\}$, each term $C_{TS}(p_0, p_i)$ is updated by Curobo [28] within 1.8s. By Alg. 2, the task plan is updated to $\Pi_{tem}\Pi_{pre} = ap_{10}ap_9ap_8ap_7ap_4$ and $\Pi_{suf} = ap_5ap_6ap_1ap_2ap_3ap_4$ within 0.27s based on its current task state. To avoid task conflicts, Alg. 3 is used to generate the set $HIN = \{\{ap_7, ap_8, ap_9\}, \{ap_7, ap_8\}, \{ap_7\}, \{\emptyset\}\}$ according to Π_{tem}, which treats the cuboid region corresponding to $L_M(ap)$ as a static obstacle when generating the control sequences. When the label $L_{mo}(m_i, o_i)$ in the environment changes during human-robot interaction, reactive planning will be performed to regenerate the path sequence $\boldsymbol{p}$ that meets the task specifications. During the interaction process, the operator's safety and the strict execution of the task specifications are guaranteed.

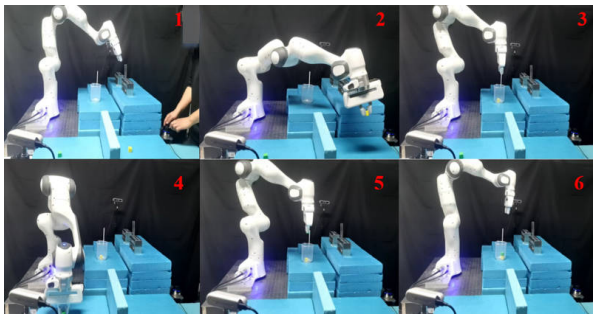

Fig. 4: The snapshots of local tasks. When the operator triggers a temporary task for the robot, the robot is able to record the current state of the global task and execute the temporary task without violating the global task constraints.

VI. CONCLUSIONS

This work develops a real-time motion planning framework that ensures the safety of human-robot interaction and enables the robot to follow a globally planned LTL task while responding to local temporary tasks. Future work will consider timed temporal logic constraints and integrate

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.

human intent recognition mechanisms to better predict and adapt to human actions in real-time.

REFERENCES

[1] Q. Zhu, F. Zhang, Y. Huang, H. Xiao, L. Zhao, X. Zhang, T. Song, X. Tang, X. Li, G. He, et al. An all-round ai-chemist with a scientific mind. *National Science Review*, 9(10), 2022.

[2] K. Darvish, M. Skreta, Y. Zhao, N. Yoshikawa, S. Som, M. Bogdanovic, Y. Cao, H. Hao, H. Xu, A. Aspuru-Guzik, et al. Organa: A robotic assistant for automated chemistry experimentation and characterization. *arXiv preprint arXiv:2401.06949*, 2024.

[3] M. Colledanchise, D. Almeida, and P. Ögren. Towards blended reactive planning and acting using behavior trees. In *2019 Proc. Int. Conf. Robot. Autom.*, pages 8839–8845. IEEE.

[4] C. Pezzato, C. H. Corbato, S. Bonhof, and M. Wisse. Active inference and behavior trees for reactive action planning and execution in robotics. *IEEE Trans. Robot.*, 39(2):1050–1069, 2023.

[5] Y. Zhang, C. Pezzato, E. Trevisan, C. Salmi, C. H. Corbato, and J. Alonso-Mora. Multi-modal mppi and active inference for reactive task and motion planning. *IEEE Rob. Autom. Lett.*, 2024.

[6] Y. Kantaros and M. M. Zavlanos. Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems. *Int. J. Robotics Res*, 39(7):812–836, 2020.

[7] X. Luo, Y. Kantaros, and M. M. Zavlanos. An abstraction-free method for multirobot temporal logic optimal control synthesis. *IEEE Trans. Robot.*, 37(5):1487–1507, 2021.

[8] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan. Modular deep reinforcement learning for continuous motion planning with temporal logic. *IEEE Robotics Autom. Lett.*, 6(4):7973–7980, 2021.

[9] M. Cai, S. Xiao, Z. Li, and Z. Kan. Optimal probabilistic motion planning with potential infeasible LTL constraints. *IEEE Trans. Autom. Control*, 68(1):301–316, 2023.

[10] Z. Chen, M. Cai, Z. Zhou, L. Li, and Z. Kan. Fast motion planning in dynamic environments with extended predicate-based temporal logic. *IEEE Trans. Autom. Sci. and Eng*, 2024.

[11] Z. Chen, Z. Zhou, S. Wang, J. Li, and Z. Kan. Fast temporal logic mission planning of multiple robots: A planning decision tree approach. *IEEE Rob. Autom. Lett.*, 2024.

[12] C. I. Vasile, X. Li, and C. Belta. Reactive sampling-based path planning with temporal logic specifications. *Int. J. Robotics Res*, 39(8):1002–1028, 2020.

[13] Y. Kantaros, M. Malencia, and G.J. Pappas. Reactive temporal logic planning for multiple robots in unknown occupancy grid maps. *arXiv preprint arXiv:2012.07912*, 2020.

[14] B. Finkbeiner, F. Klein, and N. Metzger. Live synthesis. *Innovations in Systems and Software Engineering*, 18(3):443–454, 2022.

[15] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy. Reactive task and motion planning under temporal logic specifications. In *2021 Proc. Int. Conf. Robot. Autom.*, pages 12618–12624. IEEE.

[16] Y.E. Sahin, P. Nilsson, and N. Ozay. Provably-correct coordination of large collections of agents with counting temporal logic constraints. In *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pages 249–258, 2017.

[17] V. Kurtz and H. Lin. Temporal logic motion planning with convex optimization via graphs of convex sets. *IEEE Trans. Robot.*, 39(5):3791–3804, 2023.

[18] Z. Zhou, D. J. Lee, Y. Yoshinaga, S. Balakirsky, D. Guo, and Y. Zhao. Reactive task allocation and planning for quadrupedal and wheeled robot teaming. In *Proc. IEEE 18th Int. Conf. Automat. Sci. Eng. (CASE)*, pages 2110–2117. IEEE, 2022.

[19] Z. Chen, Z. Zhou, L. Li, and Z. Kan. Active inference for reactive temporal logic motion planning. In *2024 Proc. Int. Conf. Robot. Autom.* IEEE.

[20] K. He, A. M. Wells, L. E. Kavraki, and M. Y. Vardi. Efficient symbolic reactive synthesis for finite-horizon tasks. In *2019 Proc. Int. Conf. Robot. Autom.*, pages 8993–8999. IEEE.

[21] M. Wells, Z. Kingston, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi. Finite-horizon synthesis for probabilistic manipulation domains. In *2021 Proc. Int. Conf. Robot. Autom.*, pages 6336–6342. IEEE.

[22] K. Muvvala, A. M. Wells, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi. Stochastic games for interactive manipulation domains. In *2024 Proc. Int. Conf. Robot. Autom.* IEEE.

[23] 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.

[24] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Trans. Robot.*, 34(6):1603–1622, 2018.

[25] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, and V. Isler. Ramp: Hierarchical reactive motion planning for manipulation tasks using implicit signed distance functions. In *2023 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 10551–10558. IEEE.

[26] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conf. Robot Learn.*, pages 750–759. PMLR, 2022.

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

[28] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[29] M. W. Jones, J. A. Baerentzen, and M. Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Trans. Visual. Comput. Graphics*, 12(4):581–599, 2006.

[30] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[31] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. pages 38–55, 2024.

[32] G. S. Bhat, R. Cleaveland, and A. Groce. Efficient model checking via büchi tableau automata? In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, pages 38–52. Springer, 2001.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 26,2025 at 03:14:49 UTC from IEEE Xplore. Restrictions apply.