CABTO: Context-Aware Behavior Tree Grounding for Robot Manipulation

Yishuai Cai*, Xinglin Chen*, Yunxin Mao, Kun Hu, Minglong Li†

College of Computer Science and Technology, National University of Defense Technology
{caiyishuai, chenxinglin, liminglong10}@nudt.edu.cn

Abstract

Behavior Trees (BTs) offer a powerful paradigm for designing modular and reactive robot controllers. BT planning, an emerging field, provides theoretical guarantees for the automated generation of reliable BTs. However, BT planning typically assumes that a well-designed BT system is already grounded—comprising high-level action models and low-level control policies—which often requires extensive expert knowledge and manual effort. In this paper, we formalize the BT Grounding problem: the automated construction of a complete and consistent BT system. We analyze its complexity and introduce CABTO (Context-Aware Behavior Tree grOunding), the first framework to efficiently solve this challenge. CABTO leverages pre-trained Large Models (LMs) to heuristically search the space of action models and control policies, guided by contextual feedback from BT planners and environmental observations. Experiments spanning seven task sets across three distinct robotic manipulation scenarios demonstrate CABTO's effectiveness and efficiency in generating complete and consistent behavior tree systems.

Code — https://github.com/DIDS-EI/CABTO

Introduction

Robot manipulation necessitates both reliable high-level planning and robust low-level control policies. Recently, Behavior Trees (BTs) (Ögren and Sprague 2022; Colledanchise and Ögren 2018) have emerged as a highly reliable and robust control architecture for intelligent robots, recognized for their modularity, interpretability, reactivity, and safety. Many methods have been proposed to automatically generate BTs for task execution, including evolutionary computing (Neupane and Goodrich 2019; Colledanchise, Parasuraman, and Ögren 2019) and machine learning approaches (Banerjee 2018; French et al. 2019). In particular, BT planning (Cai et al. 2021; Chen et al. 2024; Cai et al. 2025a) has shown significant promise, primarily due to its strong theoretical guarantees: the BTs generated by such methods are provably successful in achieving goals within a finite time horizon.

Despite these advancements, BT planning critically assumes the *prior existence* of a well-grounded BT system. Constructing such a system, encompassing both high-level action models and their corresponding low-level control policies, typically requires substantial human expertise and effort. Specifically, for high-level planning, the BT system must contain a sufficient and appropriately modeled set of condition and action nodes, enabling their assembly into BTs capable of accomplishing diverse tasks. Concurrently, for low-level execution, these nodes must be reliably linked to executable control policies that ensure environmental transitions occur precisely as specified by the action models, ideally with high success rates.

In this paper, we formally define the BT grounding problem: the automated construction of a complete and consistent BT system for a given task set. We characterize a well-designed BT system by two critical properties: (1) Completeness: a complete BT system can generate solution BTs for all tasks within the specified task set through high-level BT planning, based on its action models. (2) Consistency: a consistent BT system ensures that its control policies lead to state transitions that precisely match their corresponding action models during low-level BT execution. Figure 1 illustrates these concepts. For instance, a BT system with action set $\{a_2, a_3\}$ is incomplete if it can only produce a solution BT for a subset of tasks, such as $\{p_2, p_3\}$. Conversely, an action set like $\{a_1, a_2\}$ that successfully generates solution BTs for all three tasks exemplifies completeness. However, a_1 would be inconsistent if its control policy fails to achieve `Holding(apple)` as declared by its action model. Furthermore, a_2 is also inconsistent because its policy cannot put the apple in the drawer without the precondition `IsOpen(drawer)`. In contrast, an action like a_3, whose policy perfectly aligns with its action model's state transitions, is considered consistent.

We demonstrate a naive algorithm that solves the BT grounding problem via exhaustive search. While this approach effectively illustrates the core concepts, its exponential time complexity renders it impractical for deployment.

Large Models (LMs), pre-trained on extensive corpora, images or datasets in other modalities, have shown significant abilities in searching, reasoning and planning (Zhou et al. 2024; Valmeekam et al. 2023; Cai et al. 2025b). Leveraging the advantages of LMs, we propose the first frame-

*These authors contributed equally.
†Corresponding Author.

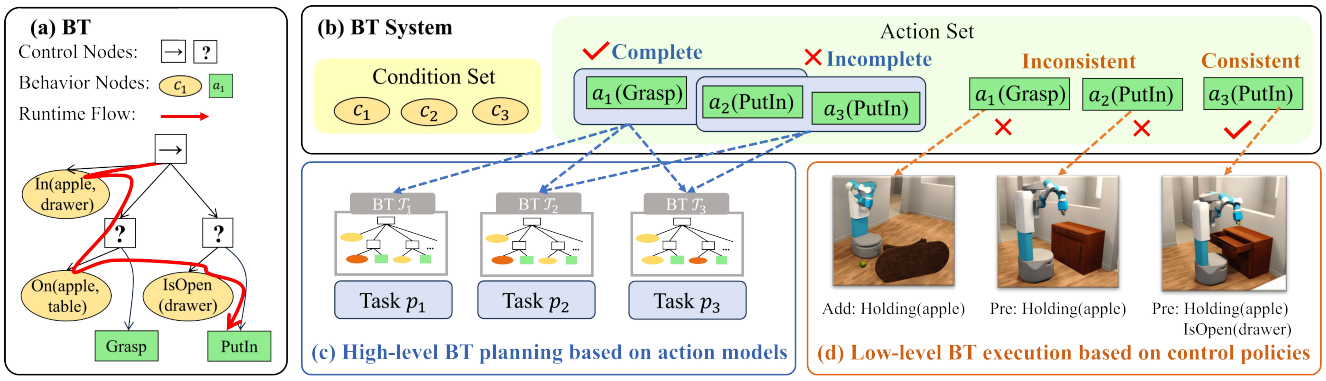


Figure 1: Concepts involved in the BT grounding problem. (a) A BT is a directed rooted tree with behavior nodes and control nodes. (b) The solution is a complete and consistent BT system for the given task set. (c) A complete BT system can generate solution BTs for all tasks during the high-level BT planning based on action models. (d) A consistent BT system ensures its control policies result in state transitions consistent with their action models during low-level BT execution.

work for efficiently solving the BT grounding problem, named Context-Aware Behavior Tree grOunding (CABTO). CABTO mainly utilizes pre-trained LMs to heuristically search the space of action models and control policies based on the contexts of BT planning and environmental feedback.

CABTO includes three phases: (1) High-level model proposal. Given a task set, we first use Large Language Models (LLMs) to generate promising action models and use a sound and complete BT planning algorithm to evaluate their completeness. The contexts here in this phase are planning details. (2) Low-level policy sampling. We then employ Vision-Language Models (VLMs) to sample promising policy types as well as their hyperparameters for explored action models. The matching policy and its corresponding action model together form a consistent action. The contexts in this phase are environment feedbacks. (3) Cross-level refinement. If the algorithm fails to find any policy for a given action model, then it is determined to be inconsistent. In this case, the contexts of both high-level planning information and low-level environment feedbacks can be combined to help VLMs to refine the action model and generate more promising action models.

The key contributions of this work are as follows:

- We formally define the BT grounding problem as the construction of a complete and consistent BT system for a given task set. We provide a formal analysis and present a naive algorithm that elucidates the foundational concepts for solving this problem.
- We propose CABTO, the first framework for efficiently solving the BT grounding problem. CABTO strategically utilizes pre-trained LMs to heuristically explore the space of action models and control policies, informed by both BT planning contexts and environmental feedback.
- We empirically validate CABTO's superior effectiveness and efficiency in automatically generating complete and consistent BT systems across 7 diverse task sets in 3 distinct robotic manipulation scenarios. Comprehensive ablation studies further investigate the impact of LMs, control policy types, and cross-level refinement.

Related Work

Behavior Tree Generation Most existing BT generation methods focus on constructing the BT structure while assuming predefined execution policies. Heuristic search approaches, including grammatical evolution (Neupane, Goodrich, and Mercer 2018), genetic programming (Lim, Baumgarten, and Colton 2010) , and Monte Carlo DAG Search (Scheide, Best, and Hollinger 2021), have been widely studied. Machine learning methods, such as reinforcement learning (Banerjee 2018; Pereira and Engel 2015)and imitation learning (French et al. 2019), as well as formal synthesis approaches like LTL (Neupane and Goodrich 2023) and its variants(Tadewos, Newaz, and Karimoddini 2022), have also been explored. However, these methods often require complex environment modeling or cannot guarantee BT reliability. In contrast, BT planning (Cai et al. 2025b; Chen et al. 2024; Cai et al. 2025a) based on STRIPS-style action models (Fikes and Nilsson 1971) provides interpretable environment modeling while ensuring both reliability and robustness.

High-Level Action Models Action models define the blueprints of actions that drive state transitions in a system (Arora et al. 2018). They are widely used in classical planning (Hoffmann and Nebel 2001; Bonet, Loerincs, and Geffner 1997), task and motion planning (TAMP) (Yang et al. 2024; Kumar et al. 2024), and symbolic problem solving (Pan et al. 2023; Fikes and Nilsson 1971). To reduce expert design effort, many methods learn action models from plan execution traces (Mahdavi et al. 2024; Bachor and Behnke 2024; Mordoch et al. 2024; Liu et al. 2023), employing inductive learning (Liang et al. 2025), evolutionary algorithms (Newton and Levine 2010), reinforcement learning (Rodrigues et al. 2012), and transfer learning (Zhuo and Yang 2014). However, these approaches typically assume the traces are already available, overlooking how to obtain them through low-level execution—an obstacle to practical deployment.

Low-Level Control Policies Modern low-level robot manipulation policies can be broadly categorized into three types: (1) End-to-end policies, which directly map proprioceptive inputs to joint controls via reinforcement learning (Bai et al. 2025; Chen et al. 2023), imitation learning (Zare et al. 2024), and, more recently, Vision-Language-Action Models (VLAs) fine-tuned from large vision-language models (Zhong et al. 2025; Kim et al. 2024; Zhen et al. 2024). (2) Hierarchical policies, which decompose control into structured modules leveraging representations such as rigid-body poses (Kaelbling and Lozano-Pérez 2011), constraints (Huang et al. 2025), affordances (Huang et al. 2023), waypoints (Zhang et al. 2024), or skills and symbolic codes (Haresh et al. 2024; Mu et al. 2024). These approaches exploit expert knowledge to improve interpretability and extend long-horizon capabilities. (3) Rule-based policies, built solely on expert-designed control algorithms (Thomason, Kingston, and Kavraki 2024; Sundaralingam et al. 2023), offer strong robustness for specific tasks but struggle to generalize to unseen scenarios.

Preliminaries

Behavior Tree A BT $\mathcal{T}$ is a rooted directed tree where internal nodes are control flow nodes and leaf nodes are execution nodes (Colledanchise and Ögren 2018). The tree is executed via periodic "ticks" from the root. The core nodes include: (1) Condition: returns `success` if a state proposition holds, else `failure`. (2) Action: performs tasks and returns `success`, `failure`, or `running`. (3) Sequence ($\rightarrow$): succeeds only if all children succeed (AND logic). (4) Fallback (?): fails only if all children fail (OR logic).

BT System Following (Cai et al. 2021), a BT can be represented as a four-tuple $\mathcal{T} = \langle n, h, \pi, r \rangle$, where n is the number of binary propositions describing the world state. Here, $h : 2^n \rightarrow 2^n$ denotes the action model representing the intended state transition; $\pi : 2^n \rightarrow 2^n$ denotes the control policy representing the actual execution effect; and $r : 2^n \mapsto \{\texttt{success}, \texttt{running}, \texttt{failure}\}$ partitions the state space according to the BT's return status.

A BT system is defined as $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$. Each action $a \in \mathcal{A}$ is a tuple $\langle h_a, \pi_a \rangle$, where $h_a = \langle pre^h(a), add^h(a), del^h(a) \rangle$ is its action model (intended effect) and $\pi_a = \langle pre^\pi(a), add^\pi(a), del^\pi(a) \rangle$ is its control policy (actual effect). The precondition $pre^h(a), pre^\pi(a)$, add effects $add^h(a), add^\pi(a)$, and delete effects $del^h(a), del^\pi(a)$ are all the subset of the condition node set $\mathcal{C}$. In a well-designed BT system, provided that the current state s_t satisfies the precondition (i.e., $s_t \supseteq pre^h(a)$), the state transition upon completion of action a after k time steps satisfies:

$$s_{t+k} = h_a(s_t) = \pi_a(s_t) = s_t \cup add(a) \setminus del(a) \quad (1)$$

where $h_a(s_t)$ and $\pi_a(s_t)$ denote the states resulting from the action model and the control policy execution, respectively.

BT Planning Given a BT system Φ, a BT planning problem is defined as: $p = \langle \mathcal{S}, s_0, g \rangle$, where $\mathcal{S}$ is the finite set of environment states, s_0 is the initial state, g is the goal condition. A condition $c \subseteq \mathcal{C}$ is a subset of a state s, and can be an atom condition node or a sequence node with atom condition nodes as children. If $c \subseteq s$, then c holds in s. A sound and complete BT planning algorithm, like BT Expansion (Cai et al. 2021), ensures a solution BT $\mathcal{T}$ in finite time if p is solvable. Such a BT $\mathcal{T}$ can transition the state from s_0 to $s_n = \pi_{\mathcal{T}}(s_0) \supseteq g$ in a finite number of steps n.

Algorithm 1: Naive Algorithm for BT Grounding

Input: Problem $\langle \mathcal{P}, \mathcal{C}_\mathcal{P}, \mathcal{H}_\mathcal{P}, \Pi_\mathcal{P} \rangle$
Output: Solution $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$

$\mathcal{A} \leftarrow \emptyset$ ▷ initialize grounded actions
for $pre \in 2^{\mathcal{C}_\mathcal{P}}, add \in 2^{\mathcal{C}_\mathcal{P}}, del \in 2^{\mathcal{C}_\mathcal{P}}$ **do**
  $h \leftarrow \langle pre, add, del \rangle$ ▷ create an action model
  if $h \in \mathcal{H}_\mathcal{P}$ **then**
    for each policy $\pi \in \Pi_\mathcal{P}$ **do**
      if `Consistent`(h, π) **then**
        $a \leftarrow \langle h, \pi \rangle$ ▷ create a consistent action
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{a\}$ ▷ add the consistent action
        break
      end if
    end for
  end if
end for
$\mathcal{C} \leftarrow \bigcup_{a \in \mathcal{A}} pre^h(a) \cup add^h(a) \cup del^h(a)$
return $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$

Problem Formulation

In this paper, we focus on the automatic construction of the BT system, and therefore need to formally define the properties that describe a well-designed BT system.

Definition 0.1 (Completeness)**.** A BT system Φ is complete in the task set $\mathcal{P}$ if, $\forall p \in \mathcal{P}$, any complete BT planning algorithm can produce a BT $\mathcal{T}$ that solves the task p according to its action models.

The completeness of the BT system Φ describes whether its condition nodes $\mathcal{C}$ and action nodes $\mathcal{A}$ are sufficient to solve all of the tasks in the given task set at the planning level.

Definition 0.2 (Consistency)**.** An action a is consistent if $pre^\pi(a) \subseteq pre^h(a), add^\pi(a) = add^h(a), del^\pi(a) = del^h(a)$. That is, the control policy π_a is capable of inducing state transitions that match its action model. A BT system Φ is consistent if $\forall a \in \mathcal{A}, a$ is consistent.

The consistency of the BT system Φ describes whether all action nodes can be successfully executed and cause the state to transition as desired, just as specified by their action models. Both completeness and consistency are essential for constructing a BT system for embodied robots to complete tasks. We then define the BT grounding problem as follows:

Problem 1 (BT Grounding). A BT grounding problem is a tuple $\langle \mathcal{P}, \mathcal{C}_\mathcal{P}, \mathcal{H}_\mathcal{P}, \Pi_\mathcal{P} \rangle$, where $\mathcal{P}$ is the finite task set, $\mathcal{C}_\mathcal{P}$ is the finite set of valid condition nodes, $\mathcal{H}_\mathcal{P}$ is the finite set of valid action models, $\Pi_\mathcal{P}$ is the set of valid control policies. A solution to this problem is a BT system $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$ that is complete and consistent in the task set $\mathcal{P}$, where $\mathcal{C} \subseteq \mathcal{C}_\mathcal{P}$, $\forall a \in \mathcal{A}, a = \langle h_a, \pi_a \rangle, h_a \in \mathcal{H} \subseteq \mathcal{H}_\mathcal{P}, \pi_a \in \Pi \subseteq \Pi_\mathcal{P}$.

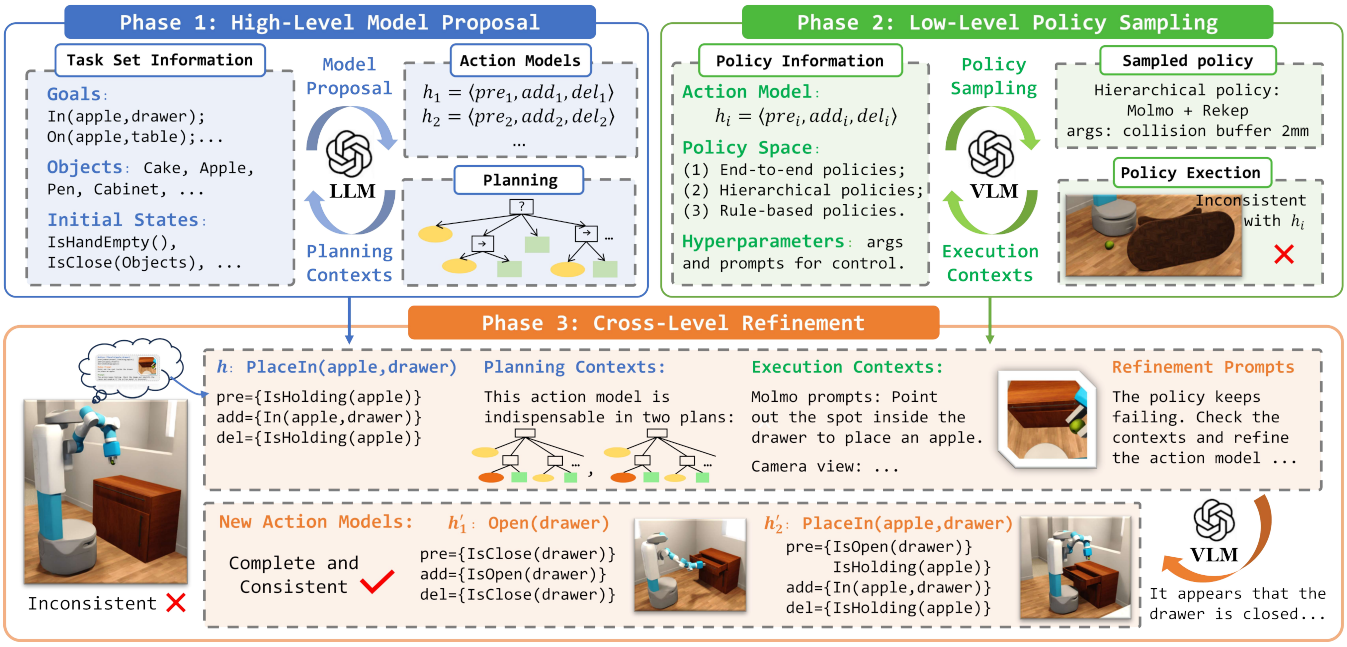


Figure 2: The framework of CABTO includes three phases: (1) High-level model proposal leverages the planning contexts for the LLMs to heuristically explore the space of action models; (2) Low-level policy sampling leverages the execution contexts for the VLMs to heuristically explore the space of control policies; (3) Cross-level refinement leverages both planning and execution contexts for refining inconsistent action models.

Methodology

This section first presents a naive algorithm and a formal analysis to establish the foundational principles of the BT grounding problem. We then detail the CABTO framework, encompassing high-level model proposal, low-level policy sampling, and cross-level refinement. Finally, we provide the implementation details of the CABTO system.

Naive Algorithm for BT Grounding Algorithm 1 outlines a naive approach to BT grounding. Given the problem tuple $\langle \mathcal{P}, \mathcal{C}_\mathcal{P}, \mathcal{H}_\mathcal{P}, \Pi_\mathcal{P} \rangle$, the algorithm initializes an empty action set $\mathcal{A}$ (line 1) and exhaustively traverses the power set of action components (line 2). For each candidate action model h, the algorithm first verifies its validity (lines 3–4). It excludes models based on domain-independent constraints (e.g., $add \cap del \neq \emptyset$) or domain-dependent constraints (e.g., mutually exclusive preconditions), though the latter typically require extensive expert knowledge. Even when restricted to domain-independent constraints, exploring the model space entails an exponential complexity of $O(2^{3n})$. The algorithm then retrieves a control policy $\pi \in \Pi_\mathcal{P}$ (line 5) and verifies its consistency with h (line 6). Upon a successful match, it instantiates a consistent action $a = \langle h, \pi \rangle$ (line 7) and appends it to $\mathcal{A}$. Finally, the algorithm induces the condition set $\mathcal{C}$ from the union of all atomic conditions in $\mathcal{A}$ (line 14) and returns the resulting BT system Φ. While exhaustive and correct, this algorithm faces significant limitations: (1) the exponential complexity of exploring $\mathcal{H}_\mathcal{P}$, and (2) the practical difficulty of designing $\Pi_\mathcal{P}$ and verifying policy consistency. Notably, automatically synthesizing low-level control policies to achieve specific effects remains a fundamental challenge in robotics (Kumar et al. 2023).

To overcome these limitations, we propose CABTO, a principled framework designed for efficient BT grounding. As illustrated in Algorithm 2, CABTO decomposes the grounding process into three phases, leveraging multimodal contexts to circumvent exhaustive search. The following sections detail the implementation and context acquisition strategies employed in each phase.

High-level Model Proposal CABTO initially initializes the grounded action set $\mathcal{A}$ as empty (Line 1) and defines the unexplored model space $\mathcal{H}_U$ as the complete set of potential action models $\mathcal{H}_\mathcal{P}$ (Line 2). The process commences with an initial proposal phase, where the LLM receives a structured textual prompt defining the task set $\mathcal{P}$. This context encapsulates goal states and initial conditions formalized as first-order logic propositions, alongside the semantics descriptions of scene objects. Leveraging this task-specific context, the LLM identifies a subset of promising models $\mathcal{H}_E$ from $\mathcal{H}_\mathcal{P}$ by specifying their symbolic preconditions and effects in a programmatic format (Line 3):

$$\mathcal{H}_E = \texttt{LLM}(\mathcal{P}, \mathcal{H}_\mathcal{P}) \tag{2}$$

Empirical results demonstrate that for simple task sets, this initial proposal phase often yields sufficient action models to satisfy the majority of requirements in $\mathcal{P}$.

To accommodate complex scenarios where initial proposals may be incomplete, CABTO employs a refinement loop that iterates until the task set $\mathcal{P}$ is verified as fully solvable using the validated grounded actions in $\mathcal{A}$ (Line 4). Within

Algorithm 2: CABTO

Input: Problem $\langle \mathcal{P}, \mathcal{C}_\mathcal{P}, \mathcal{H}_\mathcal{P}, \Pi_\mathcal{P} \rangle$
Output: Solution $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$

1: $\mathcal{A} \leftarrow \emptyset$ ▷ initialize grounded actions
2: $\mathcal{H}_U \leftarrow \mathcal{H}_\mathcal{P}$ ▷ initialize model search spaces
3: $\mathcal{H}_E \leftarrow \texttt{LLM}(\mathcal{P}, \mathcal{H}_\mathcal{P})$ ▷ Equation 2
4: **while** $\mathcal{H}_U \neq \emptyset$ **and not** $\texttt{AllSolvable}(\mathcal{P}, \mathcal{A})$ **do**
5: // **high-level model proposal**
6: **repeat**
7: $\mathcal{I}_{fail} \leftarrow \{\mathcal{I}_p \mid p \in \mathcal{P}, \texttt{BTPlanning}(p, \mathcal{H}_E) \text{ fails}\}$
8: **if** $\mathcal{I}_{fail} \neq \emptyset$ **then**
9: $\mathcal{H}' \leftarrow \texttt{LLM}(\mathcal{P}, \mathcal{H}_U, \mathcal{I}_{fail})$ ▷ Equation 3
10: $\mathcal{H}_U \leftarrow \mathcal{H}_U \setminus \mathcal{H}', \mathcal{H}_E \leftarrow \mathcal{H}_E \cup \mathcal{H}'$
11: **end if**
12: **until** $\mathcal{I}_{fail} = \emptyset$ **or** $\mathcal{H}_U = \emptyset$
13: // **low-level policy sampling**
14: **for each** $h \in \mathcal{H}_E \setminus \mathcal{H}$ **do**
15: $n \leftarrow 0, \quad \pi \leftarrow \texttt{null}, \texttt{Consistent}(h, \pi) \leftarrow \text{False}$
16: **while** $n < N_{max}$ **and not** $\texttt{Consistent}(h, \pi)$ **do**
17: $\pi \leftarrow \texttt{VLM}(h, \Pi_\mathcal{P}, \mathcal{I}_e)$ ▷ Equation 4
18: Sample a scenario s_0 where $pre(h) \subseteq s_0$
19: $s_t, \mathcal{I}_e \leftarrow \texttt{Execute}(\pi, s_0)$
20: **if** $s_t \supseteq (pre(h) \cup add(h) \setminus del(h))$ **then**
21: $\texttt{Consistent}(h, \pi) \leftarrow \text{True}$
22: $\mathcal{A} \leftarrow \mathcal{A} \cup \{\langle h, \pi \rangle\}, \mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$
23: **end if**
24: $n \leftarrow n + 1$
25: **end while**
26: // **cross-level refinement**
27: **if not** $\texttt{Consistent}(h, \pi)$ **then**
28: $h' \leftarrow \texttt{VLM}(h, \mathcal{H}_U, \{\mathcal{I}_p\}, \Pi_\mathcal{P}, \{\mathcal{I}_e\})$ ▷ Equation 5
29: $\mathcal{H}_U \leftarrow \mathcal{H}_U \setminus \{h'\}, \mathcal{H}_E \leftarrow \mathcal{H}_E \cup \{h'\}$
30: **end if**
31: **end for**
32: $\mathcal{H}_E \leftarrow \mathcal{H}$ ▷ Prune $\mathcal{H}_E$ to validated set
33: **end while**
34: $\mathcal{C} \leftarrow \bigcup_{\langle h, \pi \rangle \in \mathcal{A}} (pre(h) \cup add(h) \cup del(h))$
35: **return** $\Phi = \langle \mathcal{C}, \mathcal{A} \rangle$

this loop, the algorithm assesses the completeness of the current candidate set $\mathcal{H}_E$ by attempting to synthesize BTs for all tasks in $\mathcal{P}$ through BT Planning. Any planning failure triggers the aggregation of diagnostic data into a failure set $\mathcal{I}_{fail}$ (Line 7). Each entry $\mathcal{I}_p \in \mathcal{I}_{fail}$ encapsulates critical diagnostics, such as the topological sketches of incomplete BTs and the count of expanded conditions. These metrics provide essential semantic cues, aiding the LLM in identifying symbolic gaps to propose more promising action models.

$$\mathcal{H}' \leftarrow \texttt{LLM}(\mathcal{P}, \mathcal{H}_U, \mathcal{I}_{fail}) \tag{3}$$

Subsequently, proposed models are transferred from $\mathcal{H}_U$ to the candidate set $\mathcal{H}_E$ (Line 10). This heuristic search iterates until the task set $\mathcal{P}$ is logically spanned by a complete BT system.

Low-Level Policy Sampling This phase verifies the physical consistency of candidate action models $h \in \mathcal{H}_E$ (Line 14). For each model h, we initialize the trial counter $n = 0$ and the policy π as null. To bridge the gap between abstract symbolic reasoning and precise physical execution, we propose a hierarchical framework that integrates Molmo(Deitke et al. 2025) with programmatic code generation. Specifically, within a budget of N_{max} attempts (Line 16), the VLM is a programmatic sampler (Line 17) that translates high-level semantic intentions into grounded control policies:

$$\pi \leftarrow \text{VLM}(h, \Pi_\mathcal{P}, \mathcal{I}_e) \tag{4}$$

where $\Pi_\mathcal{P}$ represents the set of available control interfaces. These interfaces comprise Molmo-based (Deitke et al. 2024) perception APIs for extracting environmental keypoints, cuRobo-based (Sundaralingam et al. 2023) (a 7-DoF IK solver) motion control APIs for the robotic arm, and gripper actuation commands. The execution context $\mathcal{I}_e$ serves as a critical nexus for closed-loop iterative refinement. It encapsulates multi-modal diagnostic data, including egocentric visual observations, previously synthesized control code, post-hoc visual feedback, and categorical success/failure signals. By maintaining this high-fidelity temporal trace, the VLM can effectively anchor its subsequent sampling within the physical constraints evidenced by prior execution attempts.

Specifically, the VLM selectively invokes Molmo-based perception tools conditioned on the logical semantics of h. When precise spatial grounding is necessitated, the VLM leverages Molmo to extract functional affordances and task-relevant keypoints—such as optimal grasp points or target placement coordinates—directly from visual observation $\mathcal{V}$. Subsequently, the VLM synthesizes these grounded keypoints and parameterized APIs into executable Pythonic code that instantiates the specific control policy π.

To validate the policy π, we initialize a simulation s_0 such that the initial state satisfies the precondition $pre(h)$ (Line 18). After execution (Line 19), we check if the terminal state s_t achieves the expected symbolic effects: $s_t \supseteq (pre(h) \cup add(h) \setminus del(h))$ (Line 20). Upon verification, the grounded action $\langle h, \pi \rangle$ is appended to the action set $\mathcal{A}$.

Cross-Level Refinement If a sufficient number of policies fails to yield a valid policy, the action model h is deemed physically inconsistent. While a naive approach would be to discard the model and restart the high-level proposal in the next iteration, we instead leverage both planning and execution contexts to refine h (Line 28):

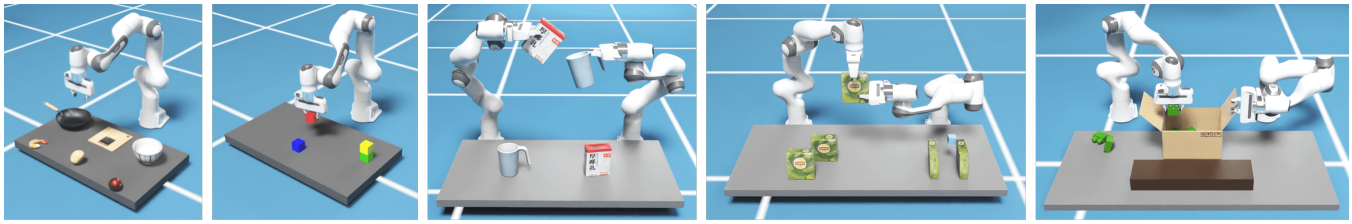
$$h' \leftarrow \texttt{VLM}(h, \mathcal{H}_U, \{\mathcal{I}_p\}, \Pi_\mathcal{P}, \{\mathcal{I}_e\}) \tag{5}$$

Here, the VLM synthesizes the planning context $\{\mathcal{I}_p\}$, which defines the functional necessity of h within successful symbolic sequences ($\forall \mathcal{I}_p \in \{\mathcal{I}_p\}, h \in \mathcal{T}_p$), and the execution context $\{\mathcal{I}_e\}$, which comprises multi-modal diagnostic data such as egocentric pre/post-action imagery and binary feedback. By integrating these cross-level insights, the VLM identifies underlying failures—such as omitted spatial preconditions or inaccurate symbolic effects—to synthesize a rectified action model h'.

Upon completing the refinement loop, a knowledge synchronization step (Line 32) updates $\mathcal{H}_E \leftarrow \mathcal{H}$, ensuring the explored pool consists exclusively of models verified by physical policies. This update provides a grounded, reliable action library for subsequent planning iterations (Line 7). The cycle repeats until a set of grounded actions $\mathcal{A}$ renders all tasks $\mathcal{P}$ solvable, after which the condition set $\mathcal{C}$ is extracted to define the final grounded state space (Line 34).

Robot	Task Set	Task Attributes			GPT-3.5-Turbo			GPT-4o		
		Acts	Conds	Steps	ASR(w/o → w)	CSR(w/o → w)	FC	ASR(w/o → w)	CSR(w/o → w)	FC
Franka	**Cover**	2.0	4.4	4.0	60.0% → **66.7%**	40% → **50%**	1.6	100.0% → **100.0%**	100% → **100%**	0.0
	Blocks	2.3	3.1	4.1	70.0% → **70.0%**	30% → **50%**	2.0	60.0% → **80.0%**	50% → **80%**	1.1
Dual-Franka	**Pour**	5.5	8.1	3.6	80.0% → **96.7%**	70% → **90%**	0.5	66.7% → **100.0%**	60% → **100%**	0.6
	Handover	5.0	3.3	2.7	80.0% → **90.0%**	70% → **90%**	0.5	56.7% → **90.0%**	30% → **90%**	1.3
	Storage	6.0	5.4	2.2	56.7% → **73.3%**	0% → **60%**	2.0	53.3% → **76.7%**	20% → **70%**	1.7
Fetch	**Tidy Home**	5.6	6.0	2.9	53.3% → **56.7%**	40% → **50%**	0.7	53.3% → **90.0%**	30% → **90%**	1.3
	Cook Meal	6.8	7.9	5.1	70.0% → **70.0%**	50% → **60%**	0.7	73.3% → **100.0%**	60% → **100%**	0.4
Total		4.7	5.5	3.5	67.1% → **74.8%**	42.9% → **64.3%**	1.1	66.2% → **91.0%**	50% → **90.0%**	0.9

Table 1: High-level model proposal results (averaged over 10 trials, max FC=3) for GPT-3.5-turbo vs. GPT-4o. Note: "w/o" denotes without planning contexts, and "w" denotes with planning contexts.



(1) Cover (2) Blocks (3) Pour (4) Handover (5) Storage

Figure 3: Configurations of the single-arm and dual-arm Franka manipulation tasks in Isaac Sim.

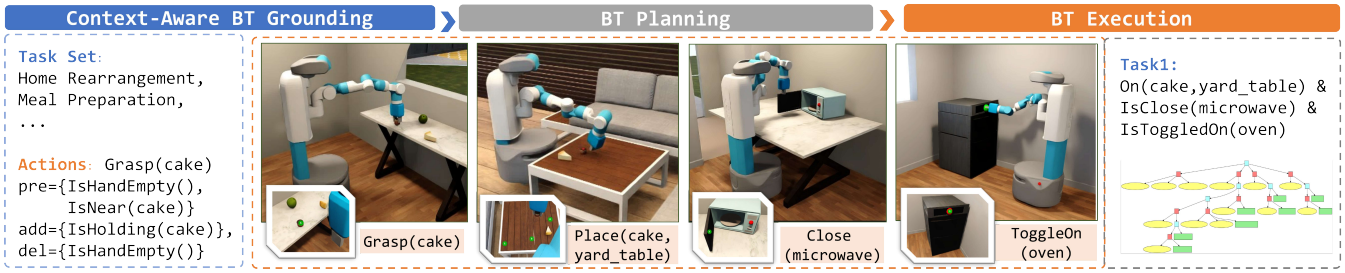


Figure 4: The deployment of CABTO in OmniGibson: Given a task set, CABTO generates a complete and consistent BT system. For a specific task, BT planning is used to generate the solution BT. Then the BT is executed, enabling the robot to successfully achieve the goal.

Experimental Setup

Task Sets We evaluate the robustness and adaptability of CABTO on a comprehensive suite of seven robotic manipulation task sets, encompassing 21 unique goals (three goals per task) across three distinct robotic platforms. These scenarios are strategically designed to cover a spectrum of physical and logical challenges: Single-Arm Franka (T1: Cover, T2: Blocks), Dual-Arm Franka (T3: Pour, T4: Handover, T5: Storage), and Mobile Fetch (T6: Tidy Home, T7: Cook Meal). As summarized in Table 1, these tasks range from fundamental pick-and-place and stacking (T1–T2) to complex bimanual coordination for cooperative transport and exchange (T3–T5), and long-horizon mobile manipulation involving articulated objects and semantic state changes (T6–T7). To quantify solution complexity, Table 1 reports the resulting BT attributes for each task set, including the number of unique action predicates (Acts), condition predicates (Conds), and the total execution steps (Steps).

Environment Fetch robot experiments were conducted in OmniGibson (Li et al. 2023) for its realistic physics, while Franka tasks were designed in Isaac Sim to enable flexible object configuration (Figures 3 and Figures 4). All experiments are conducted on a single NVIDIA RTX 4090 GPU.

Metrics We evaluate the completeness of the high-level model using two primary metrics: (1) Average Planning Success Rate (ASR): The mean planning success rate across all individual tasks within a given task set. (2) Complete Planning Success Rate (CSR): The success rate where all tasks within the set are successfully planned simultaneously. We also report the average number of Feedback Cycles (FC).

Action	End-to-end	Hierarchical			Rule-based	Molmo+cuRobo+APIs	
	OpenVLA	VoxPoser	ReKep	Molmo+cuRobo	APIs	w/o Contexts	with Contexts
`Pick`(obj)	4/10	4/10	6/10	5/10	6/10	6/10	**7/10**
`Place`(obj, loc)	5/10	3/10	7/10	5/10	6/10	6/10	**8/10**
`Open`($container$)	1/10	1/10	1/10	3/10	1/10	2/10	**4/10**
`Close`($container$	2/10	2/10	3/10	4/10	2/10	3/10	**5/10**
`Toggle`($switch$)	2/10	1/10	4/10	6/10	5/10	5/10	**7/10**
Total	28%	22%	42%	46%	40%	44%	**62%**

Table 2: Evaluation results of low-level policy sampling using VLM for 5 typical action models.

Action	Defect Type & Description	Textual Baseline	w/o Feedback	with Feedback	Avg. FC
`PutIn`($obj, container$)	**Pre:** Missing `IsOpen`($container$) due to closed lid	10%	40%	**80%**	1.1
`Stack`(obj_a, obj_b)	**Pre:** Missing `Clear`(obj_b) due to surface obstruction	20%	30%	**70%**	2.1
`Lift`(box_{big}, r_1, r_2)	**Pre:** Missing `Holding`(r_2, box_{big}) in dual-arm coordination	10%	80%	**90%**	0.3
`Pick`($robot, obj$)	**Add:** Unverified `InReach`($robot, obj$) (Kinematic constraint)	20%	50%	**90%**	0.8
`Put`(obj, loc)	**Del:** Stale `At`(obj, loc_{old}) resulting in location redundancy	0%	20%	**40%**	2.4
Total		12%	44%	**74%**	1.3

Table 3: Success rate (SR%) of VLM-based cross-level refinement for action models. Results are averaged over 10 trials ($N_{FC} \leq 3$). **Pre**, **Add**, and **Del** represent action precondition, add effect, and delete effect, respectively.

Evaluation of High-Level Model Proposal

Ablating Planning Contexts Planning context feedback proved crucial for performance (Table 1). Its inclusion consistently boosted goal success rates and system completeness, most notably for GPT-4o, where completeness jumped from 50% to over 90%. The performance gains were most significant in the complex dual-arm and mobile manipulation tasks, demonstrating that structured, symbolic feedback from a formal planner can empower LLMs to resolve intricate logical challenges.

Comparison of LLMs As shown in Table 1, while GPT-3.5 and GPT-4o performed comparably without planning context feedback, GPT-4o's superiority became evident with it. Guided by this feedback, GPT-4o achieved over 90% complete planning success rate, in stark contrast to approximately 60% for GPT-3.5. This underscores GPT-4o's advanced capacity for leveraging contextual feedback in complex reasoning tasks like BT grounding.

Evaluation of Low-Level Policy Sampling

We evaluate the performance of three policy types for low-level policy sampling. Details of these policy are shown in Appendix. We select five typical action models to test the performance of these polices, as shown in Table 2. The algorithms show different strengths in various actions. Rekep and Rule-Based methods excel in grasping, while Molmo+cuRobo performs better in `Open`/`Close` and `Toggle` actions. This is due to the semantics-based keypoint extraction that accurately identifies object handles and hinges. We utilize GPT-4o as VLM in the experiment.

Ablating Execution Contexts Table 2 presents the Success Rate (SR) of control policy for five typical action models, where the VLM samples the policy type and its hyperparameters based on the execution contexts. The results show the SR without execution contexts and with up to three sample attempts. It is evident that the VLM can effectively sample low-level policies, and with execution contexts, the SR of the actions improves.

Evaluation of Cross-Level Refinement

Ablating Environment Feedback Table 3 catalogs action models that exhibited inconsistencies, where the predicted high-level effect diverged from the low-level execution outcome or resulted in an error. Through an iterative feedback process, the VLM demonstrated the potential to successfully correct these high-level representations, underscoring the critical role of direct environmental feedback. However, the efficacy of this approach is currently limited for abstract concepts lacking direct visual correlates, such as the symbolic target in `Put`(obj, loc).

Deployment Figure 4 depicts the deployment of our pipeline, where CABTO generates a complete and consistent BT system for the given task set. The robot successfully executes the planned BT actions sequentially for every task.

Conclusion

In this work, we first formalize the BT grounding problem and propose CABTO, a framework that leverages LMs to automatically construct complete and consistent BT systems guided by planning and environmental feedback. The effectiveness of our approach is validated across seven robotic manipulation task sets. Future work will focus on enhancing LM inference and low-level robotic skills via fine-tuning and addressing the transfer to physical systems.

Acknowledgments

The authors thank Yaodong Yang (Peking University) and Yuanpei Chen (PsiBot) for their helpful insights and platform support regarding the robotic strategy implementation. This work was supported by the National Science Fund for Distinguished Young Scholars (Grant Nos. 62525213), the National Natural Science Foundation of China (Grant Nos. 62572480), and the University Youth Independent Innovation Science Foundation (Grant Nos. ZK25-11).

References

Arora, A.; Fiorino, H.; Pellier, D.; Métivier, M.; and Pesty, S. 2018. A Review of Learning Planning Action Models. *The Knowledge Engineering Review*, 33: e20.

Bachor, P.; and Behnke, G. 2024. Learning Planning Domains from Non-Redundant Fully-Observed Traces: Theoretical Foundations and Complexity Analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 20028–20035.

Bai, F.; Li, Y.; Chu, J.; Chou, T.; Zhu, R.; Wen, Y.; Yang, Y.; and Chen, Y. 2025. Retrieval dexterity: Efficient object retrieval in clutters with dexterous hand. *arXiv preprint arXiv:2502.18423*.

Banerjee, B. 2018. Autonomous Acquisition of Behavior Trees for Robot Control. 3460–3467.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A Robust and Fast Action Selection Mechanism for Planning. In *AAAI*, 714–719.

Cai, Y.; Chen, X.; Cai, Z.; Mao, Y.; Li, M.; Yang, W.; and Wang, J. 2025a. Mrbtp: Efficient multi-robot behavior tree planning and collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 14548–14557.

Cai, Y.; Chen, X.; Mao, Y.; Li, M.; Yang, S.; Yang, W.; and Wang, J. 2025b. HBTP: Heuristic Behavior Tree Planning with Large Language Model Reasoning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 13706–13713.

Cai, Z.; Li, M.; Huang, W.; and Yang, W. 2021. BT Expansion: A Sound and Complete Algorithm for Behavior Planning of Intelligent Robots with Behavior Trees. In *AAAI*, 6058–6065. AAAI Press.

Chen, X.; Cai, Y.; Mao, Y.; Li, M.; Yang, W.; Xu, W.; and Wang, J. 2024. Integrating Intent Understanding and Optimal Behavior Planning for Behavior Tree Generation from Human Instructions. In *IJCAI*.

Chen, Y.; Wang, C.; Fei-Fei, L.; and Liu, C. K. 2023. Sequential Dexterity: Chaining Dexterous Policies for Long-Horizon Manipulation. *arXiv preprint arXiv:2309.00987*.

Colledanchise, M.; and Ögren, P. 2018. *Behavior Trees in Robotics and AI: An Introduction*. CRC Press.

Colledanchise, M.; Parasuraman, R.; and Ögren, P. 2019. Learning of Behavior Trees for Autonomous Agents. *IEEE Transactions on Games*, 11(2): 183–189.

Deitke, M.; Clark, C.; Lee, S.; Tripathi, R.; Yang, Y.; Park, J. S.; Salehi, M.; Muennighoff, N.; Lo, K.; Soldaini, L.; et al. 2024. Molmo and Pixmo: Open Weights and Open Data for State-of-the-Art Multimodal Models. *arXiv preprint arXiv:2409.17146*.

Deitke, M.; Clark, C.; Lee, S.; Tripathi, R.; Yang, Y.; Park, J. S.; Salehi, M.; Muennighoff, N.; Lo, K.; Soldaini, L.; et al. 2025. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 91–104.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. 2(3-4): 189–208.

French, K.; Wu, S.; Pan, T.; Zhou, Z.; and Jenkins, O. C. 2019. Learning Behavior Trees from Demonstration. In *2019 International Conference on Robotics and Automation (ICRA)*, 7791–7797. IEEE.

Haresh, S.; Dijkman, D.; Bhattacharyya, A.; and Memisevic, R. 2024. ClevrSkills: Compositional Language And Visual Reasoning in Robotics.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

Huang, W.; Wang, C.; Li, Y.; Zhang, R.; and Fei-Fei, L. 2025. ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation. In *Conference on Robot Learning*, 4573–4602. PMLR.

Huang, W.; Wang, C.; Zhang, R.; Li, Y.; Wu, J.; and Fei-Fei, L. 2023. Voxposer: Composable 3d Value Maps for Robotic Manipulation with Language Models. *arXiv preprint arXiv:2307.05973*.

Kaelbling, L. P.; and Lozano-Pérez, T. 2011. Hierarchical Task and Motion Planning in the Now. In *2011 IEEE International Conference on Robotics and Automation*, 1470–1477. IEEE.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; et al. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*.

Kumar, N.; Ramos, F.; Fox, D.; and Garrett, C. R. 2024. Open-World Task and Motion Planning via Vision-Language Model Inferred Constraints. *arXiv preprint arXiv:2411.08253*.

Kumar, V.; Shah, R.; Zhou, G.; Moens, V.; Caggiano, V.; Gupta, A.; and Rajeswaran, A. 2023. RoboHive: A Unified Framework for Robot Learning. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 44323–44340. Curran Associates, Inc.

Li, C.; Zhang, R.; Wong, J.; Gokmen, C.; Srivastava, S.; Martín-Martín, R.; Wang, C.; Levine, G.; Lingelbach, M.; Sun, J.; Anvari, M.; Hwang, M.; Sharma, M.; Aydin, A.; Bansal, D.; Hunter, S.; Kim, K.-Y.; Lou, A.; Matthews, C. R.; Villa-Renteria, I.; Tang, J. H.; Tang, C.; Xia, F.; Savarese, S.; Gweon, H.; Liu, K.; Wu, J.; and Fei-Fei, L.

2023. BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation. In Liu, K.; Kulic, D.; and Ichnowski, J., eds., *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, 80–93. PMLR.

Liang, Y.; Kumar, N.; Tang, H.; Weller, A.; Tenenbaum, J. B.; Silver, T.; Henriques, J. F.; and Ellis, K. 2025. VisualPredicator: Learning Abstract World Models with Neuro-Symbolic Predicates for Robot Planning. In *The Thirteenth International Conference on Learning Representations*.

Lim, C.-U.; Baumgarten, R.; and Colton, S. 2010. Evolving Behaviour Trees for the Commercial Game DEFCON. In *Applications of Evolutionary Computation: EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I*, 100–110. Springer.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv*.

Mahdavi, S.; Aoki, R.; Tang, K.; and Cao, Y. 2024. Leveraging Environment Interaction for Automated PDDL Generation and Planning with Large Language Models. *arXiv preprint arXiv:2407.12979*.

Mordoch, A.; Scala, E.; Stern, R.; and Juba, B. 2024. Safe Learning of Pddl Domains with Conditional Effects. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 387–395.

Mu, Y.; Chen, J.; Zhang, Q.; Chen, S.; Yu, Q.; Ge, C.; Chen, R.; Liang, Z.; Hu, M.; Tao, C.; et al. 2024. RoboCodeX: Multimodal Code Generation for Robotic Behavior Synthesis. *arXiv preprint arXiv:2402.16117*.

Neupane, A.; and Goodrich, M. 2019. Learning Swarm Behaviors Using Grammatical Evolution and Behavior Trees. In *IJCAI*, 513–520. IJCAI Organization.

Neupane, A.; and Goodrich, M. A. 2023. Designing Behavior Trees from Goal-Oriented LTLf Formulas. *arXiv preprint arXiv:2307.06399*.

Neupane, A.; Goodrich, M. A.; and Mercer, E. G. 2018. GEESE: Grammatical Evolution Algorithm for Evolution of Swarm Behaviors. 999–1006.

Newton, MA.; and Levine, J. 2010. Implicit Learning of Compiled Macro-Actions for Planning. In *ECAI 2010*, 323–328. IOS Press.

Ögren, P.; and Sprague, C. I. 2022. Behavior Trees in Robot Control Systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 5: 81–107.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. Y. 2023. Logic-Lm: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. *arXiv preprint arXiv:2305.12295*.

Pereira, R. d. P.; and Engel, P. M. 2015. A Framework for Constrained and Adaptive Behavior-Based Agents. *arXiv preprint arXiv:1506.02312*.

Rodrigues, C.; Gérard, P.; Rouveirol, C.; and Soldano, H. 2012. Active Learning of Relational Action Models. In *Inductive Logic Programming: 21st International Conference, ILP 2011, Windsor Great Park, UK, July 31–August 3, 2011, Revised Selected Papers 21*, 302–316. Springer.

Scheide, E.; Best, G.; and Hollinger, G. A. 2021. Behavior Tree Learning for Robotic Task Planning through Monte Carlo DAG Search over a Formal Grammar. 4837–4843. Xi'an, China: IEEE. ISBN 978-1-72819-077-8.

Sundaralingam, B.; Hari, S. K. S.; Fishman, A.; Garrett, C.; Van Wyk, K.; Blukis, V.; Millane, A.; Oleynikova, H.; Handa, A.; Ramos, F.; et al. 2023. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 8112–8119. IEEE.

Tadewos, T. G.; Newaz, A. A. R.; and Karimoddini, A. 2022. Specification-Guided Behavior Tree Synthesis and Execution for Coordination of Autonomous Systems. *Expert Systems with Applications*, 201: 117022.

Thomason, W.; Kingston, Z.; and Kavraki, L. E. 2024. Motions in Microseconds via Vectorized Sampling-Based Planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 8749–8756.

Valmeekam, K.; Marquez, M.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 38975–38987. Curran Associates, Inc.

Yang, Z.; Garrett, C.; Fox, D.; Lozano-Pérez, T.; and Kaelbling, L. P. 2024. Guiding Long-Horizon Task and Motion Planning with Vision Language Models. *arXiv preprint arXiv:2410.02193*.

Zare, M.; Kebria, P. M.; Khosravi, A.; and Nahavandi, S. 2024. A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges. *IEEE Transactions on Cybernetics*.

Zhang, K.; Ren, P.; Lin, B.; Lin, J.; Ma, S.; Xu, H.; and Liang, X. 2024. PIVOT-R: Primitive-Driven Waypoint-Aware World Model for Robotic Manipulation. *arXiv preprint arXiv:2410.10394*.

Zhen, H.; Qiu, X.; Chen, P.; Yang, J.; Yan, X.; Du, Y.; Hong, Y.; and Gan, C. 2024. 3d-Vla: A 3d Vision-Language-Action Generative World Model. *arXiv preprint arXiv:2403.09631*.

Zhong, Y.; Bai, F.; Cai, S.; Huang, X.; Chen, Z.; Zhang, X.; Wang, Y.; Guo, S.; Guan, T.; Lui, K. N.; et al. 2025. A Survey on Vision-Language-Action Models: An Action Tokenization Perspective. *arXiv preprint arXiv:2507.01925*.

Zhou, A.; Yan, K.; Shlapentokh-Rothman, M.; Wang, H.; and Wang, Y.-X. 2024. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 62138–62160. PMLR.

Zhuo, H. H.; and Yang, Q. 2014. Action-Model Acquisition for Planning via Transfer Learning. *Artificial intelligence*, 212: 80–103.