2025 IEEE International Conference on Robotics and Automation (ICRA)
May 19-23, 2025. Atlanta, USA

Trustworthy Robot Behavior Tree Generation based on Multi-source Heterogeneous Knowledge Graph

Jianchao Yuan[1], Shuo Yang[1*], Qi Zhang[1], Ge Li[1], Jianping Tang[1]

***Abstract*—In robotics, the design of robot behavior trees generally requires roboticists to comprehensively and customizable consider all the relevant factors including the robot hardware capabilities, task descriptions, etc, posing great challenges for design quality and efficiency. The mainstream practice of BT design paradigm has been utilizing the BT component framework to develop task-specific BT structures manually. In contrast, the latest advances in Generative Pretrained Transformers (GPTs) have also opened up the possibility of BT design automation. However, these approaches generally show low efficiency or are less trustworthy for complex robot task goals due to time-consuming manual design and unreliable GPT reasoning. To solve the above limitations, this paper proposes a novel knowledge-driven approach that develops a specialized knowledge graph from multi-sourced and heterogeneous high-quality robot knowledge to reason out a trustworthy robot plan for achieving complex task goals. Then we present the plan transformation and BT merging algorithms to automatically generate the plan-level BT structure. The comparative experiment results have shown that our approach can generate high-quality and trustworthy BT structure regarding the task plan accuracy and consistency, as well as the BT generation time, compared with the manual design and GPT-based approaches.**

I. Introduction

In recent years, Behavior Trees (BTs) have gained significant popularity as an effective robotic software architecture for addressing task requirements [1], [2]. BTs provide a structured framework that translates robot task plans into executable actions [3]. Typically, BT design requires roboticists to interpret ambiguous natural language task descriptions, evaluate robot platform capabilities and environmental factors, and generate a suitable BT structure to meet task requirements [4]. However, as tasks become more complex, ensuring BT design quality and efficiency has become increasingly challenging [5].

In recent years, the robotic research community has primarily relied on manual behavior tree (BT) design using component-based development frameworks like BehaviorTree.CPP [6]. Roboticists typically design BT structures using their expertise and reusing open-source components. However, relying on personal expertise can lead to low-quality or error-prone designs, especially for complex tasks requiring substantial time and effort. More recently, Generative Pre-trained Transformers (GPTs) have been introduced for designing and generating BTs by reasoning through task descriptions and creating plausible BTs. Despite their potential, the inherent probabilistic nature of GPT models often leads to inconsistent and unreliable BT designs, with significant variation in results for the same task requirements. This unpredictability and lack of trustworthiness make GPT-based approaches unsuitable for safety-critical robotic tasks.

To tackle the above limitations, this paper aims to automate the design process of robot behavior trees efficiently and reliably. With the rise of the open-source movement, numerous high-quality robot knowledge models have emerged, though they exist in heterogeneous forms. These knowledge models, abstracted and designed by expert groups, provide trustworthy solutions to common robot task requirements. Knowledge Graphs (KGs) offer a powerful method for modeling the properties and relationships of entities, enabling robots to make complex inferences and apply these solutions effectively in task planning.

Integrating multi-source heterogeneous knowledge with behavior tree (BT) architectures provides a robust approach for generating robot plans, combining symbolic reasoning with modular control logic. BTs offer a clear and flexible representation of complex control due to their modular, hierarchical structure. Unlike finite state machines or rule-based systems, BTs simplify debugging, maintenance, and component reuse, enhancing scalability. Their rich control flow, including sequences, selectors, and parallel executions, allows precise task execution and real-time adaptation, ensuring that human operators interpret and modify plans.

To fully leverage valuable knowledge resources and address these challenges, this paper proposes a novel BT design approach using multi-source heterogeneous robotic knowledge to efficiently construct an explainable knowledge graph and generate trustworthy robot BTs. The approach reduces reliance on expert experience, improving reproducibility and accuracy. Additionally, the BT design's efficiency and trustworthiness are evaluated based on task plan accuracy, consistency, and generation time.

II. Related Work

A. ROS-based Behavior Tree Generation

Behavior Trees (BTs) are widely utilized in the ROS framework for their modularity and flexibility in managing

[1]State Key Laboratory of Digital-Intelligent Modeling and Simulation, College of Systems Engineering, National University of Defense Technology, Changsha, China

This work is supported by National Natural Science Youth Fund Project(Fund No. 62103420) and National University of Defense Technology Young Scientist Self-Innovation Science Fund Project(ZK2023-36)

*Corresponding author: yangshuo11@nudt.edu.cn

2025 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3315-4139-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICRA55743.2025.11127406

complex tasks. However, manual component-based generation is time-consuming and heavily reliant on expert input, limiting efficiency. S. Macenski *et al.* demonstrate BTs' effectiveness in navigation, yet manual configuration remains labor-intensive [7]. J. Á. Segura-Muros and J. Fernández-Olivares show that even with hierarchical task networks (HTNs), BTs still depend on expert-defined tasks, hindering adaptation to environmental changes [8].

A. Joon and W. Kowalczyk highlight BTs' complexity in multi-robot coordination, making manual construction error-prone and less scalable [9]. T. Ribeaud and C. Zhang Sprenger benefit from ROS2's real-time features but still face adaptability challenges [10]. S. Yang *et al.* enhances adaptability with observation mechanisms, but reliance on predefined nodes limits flexibility [11]. Z. Han underscores the limitations of expert-driven BT systems, particularly in achieving adaptive behavior [12]. In summary, while ROS-based BTs are modular and reusable, their manual nature creates inefficiencies, limiting scalability and responsiveness.

B. GPT-based Behavior Tree Generation

GPT-based behavior tree (BT) generation enhances efficiency by using large language models (LLMs) to translate natural language commands into BTs, reducing the need for expert input. M. G. Arenas *et al.* and L. Sun *et al.* show how LLMs expedite task planning by directly generating executable sequences [13], [14].

However, accuracy and consistency pose challenges. G. Tziafas and H. Zhou *et al.* highlight how LLM outputs can vary, causing inconsistencies and errors [15], [16]. Y. Chen *et al.* show increased complexity with multi-robot tasks, while Z. Zhou *et al.* emphasize the need for iterative refinements to correct LLM-generated plans [17], [18]. LLMs can misinterpret context in dynamic environments, resulting in imprecise BTs [19].

In summary, GPT-based methods offer speed and flexibility but struggle with trustworthiness, accuracy, and consistency.

C. Knowledge Graph for Robotics

Knowledge graphs (KGs) enhance decision-making, autonomy, and adaptability in robotics. E. Bartoli*et al.* used KG embeddings to incrementally build a robot's knowledge base for long-term interaction [20]. Wilcock and Jokinen integrated KGs with conversational AI to improve context-aware responses in social robots [21].

Zhou *et al.* combined KGs with behavior trees to increase task flexibility [22], while Song *et al.* introduced an encoder-decoder model for KG completion to improve task planning accuracy [23]. Xu *et al.* tackled the challenge of integrating heterogeneous knowledge sources [24], and Zhao *et al.* enhanced ROS modularity through KG-based ROS node construction [25]. Miao *et al.* applied KGs for semantic robot manipulation [26].

KGs also support trustworthiness and explainability. Daruna *et al.* proposed explainable KG inferences for trustworthy robot actions [27], while Deng *et al.* used KGs for fault diagnosis in complex systems [28]. Wilcock and Jokinen highlighted the importance of trustworthiness in KG-derived information [29].

III. Multi-source Heterogeneous Knowledge Graph Construction

A. Overview

This section outlines the construction of our task-level robot knowledge graph by merging multi-source knowledge bases with diverse representations. The selected bases include KnowRob[1] from the University of Bremen, which models long-term robot knowledge and actions [30], and SOMA[2], which encodes human-robot interactions in socio-technical systems [31]. Fig. 1 shows the process of constructing the pattern and data layers of the knowledge graph.

B. Pattern Layer: Conceptual Robotic Tasks and Actions

As shown in Fig. 2, KnowRob and SOMA employ distinct syntactic structures—KnowRob uses a **Procedural** approach, representing tasks as sequences of actions. At the same time, SOMA follows a **Declarative** approach, focusing on conditions and outcomes. These differences, while practical for their respective systems, complicate knowledge integration. To address this, the pattern layer of the knowledge graph provides a unified conceptual schema, standardizing high-level robot task and action representations.

1) Concept Extraction and Alignment: The first step in constructing the Pattern Layer involves extracting and aligning key task and action concepts from KnowRob and SOMA. Concepts like "PickUpObject" (KnowRob) and "GraspObject" (SOMA) are compared based on semantic similarity, measured using **Cosine Similarity**. Each concept is represented as a vector (e.g., word embeddings), and the cosine similarity between two vectors A and B is:

$$\text{Cosine Similarity} = \frac{A \cdot B}{\|A\|\|B\|} \tag{1}$$

where $A \cdot B$ is the dot product, and $\|A\|$ and $\|B\|$ are the magnitudes. A threshold of 0.8 balances capturing strong semantic similarity while allowing for minor differences. Concepts scoring above 0.8 are aligned and treated as equivalent, forming a unified representation in the Pattern Layer.

2) Abstracting High-level Concepts: After alignment, the concepts are abstracted into generalized, high-level representations. For instance, "PickUpObject" and "GraspObject" are abstracted into the broader concept of "ObjectManipulation." This creates standardized concepts independent of the syntactic structures of the original knowledge bases and represented as universal entities in the knowledge graph.

3) Defining Nodes and Relationships: The abstracted concepts are translated into nodes within the knowledge graph, representing generalized tasks or actions like "ObjectManipulation." Relationships between these nodes define the procedural and conditional logic.

[1]https://www.knowrob.org/

[2]https://ease-crc.github.io/soma/

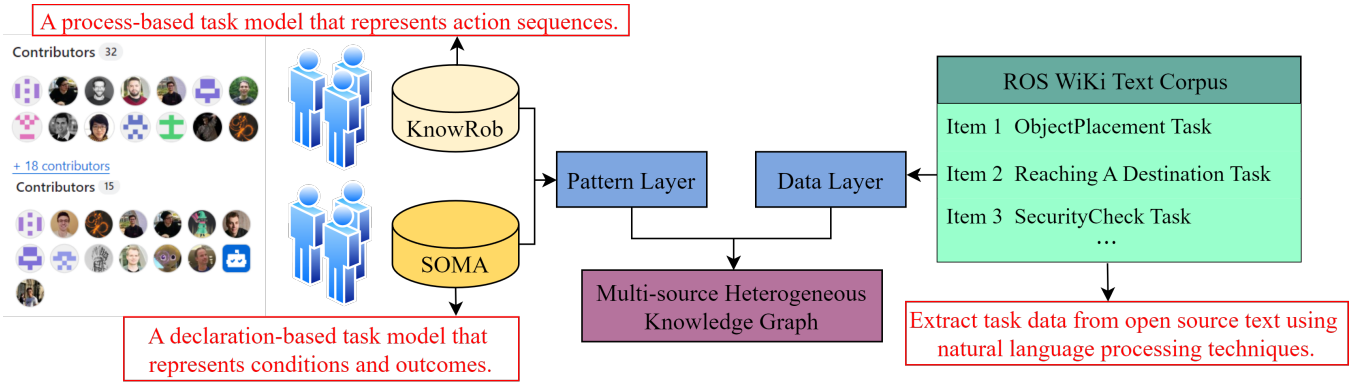

Fig. 1: The Construction of Multi-source Heterogeneous Knowledge Graph

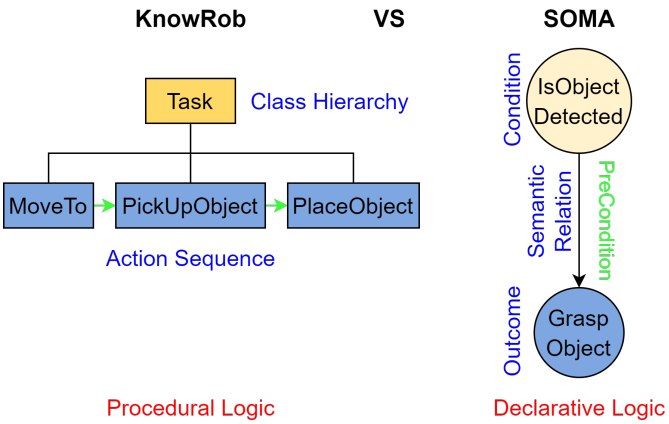

Fig. 2: A comparison of the syntactic and semantic differences between KnowRob and SOMA.

4) Mapping Rules and Transformation Algorithm: To integrate concepts from KnowRob and SOMA into the knowledge graph, **mapping rules** are defined. These rules specify how tasks and actions map to nodes and relationships.

The **mapping rules** include:

- Mapping concepts from KnowRob and SOMA to high-level nodes.
- Defining how actions, tasks, and conditions correspond to nodes and edges.
- Setting conditions for matching, based on task names or attributes.

To automate this process, Algorithm 1 applies these rules, parsing concepts from KnowRob and SOMA, and generating corresponding nodes and edges in the **M**ulti-source **H**eterogeneous **K**nowledge **G**raph (**MHKG**):

Core variables:

- `knowrob_graph` and `soma_graph`: Parsed OWL files.
- `MHKG_graph`: The merged multi-source knowledge graph.
- `MHKG_node`: Node in the MHKG corresponding to the mapped concept.
- `MapConcept`: Determines equivalent high-level concept in MHKG.

Algorithm 1 Concept Mapping to Multi-source Heterogeneous Knowledge

Input: KnowRob and SOMA OWL files
Output: Multi-source Heterogeneous Knowledge Graph (MHKG.owl)
Load and parse OWL files: `knowrob_graph`, `soma_graph`
Initialize `MHKG_graph` with namespace `MHKG_ns`
for all *concept* in `knowrob_graph.subjects()` + `soma_graph.subjects()` **do**
    if *concept* = "PickUpObject" or *concept* = "GraspObject" **then**
        `MHKG_node` ← `MHKG_ns.ObjectManipulation`
    end if
    if `MHKG_node` exists **then**
        `MHKG_graph.add(MHKG_node, MHKG_ns.hasTask, concept)` **or** `MHKG_ns.hasAction`
    end if
end for
Save `MHKG_graph` as "MHKG.owl"

C. Data Layer: Open Robot Textual Information Retrieval

Building on the constructed Pattern Layer, the Data Layer extracts relevant keywords from the ROS Wiki Text corpus. This is crucial for populating the Multi-source Heterogeneous Knowledge with data that mirrors real-world robotic tasks and actions. By using advanced NLP techniques, key concepts are extracted from unstructured text and aligned with the standardized representations in the Pattern Layer.

1) Text Corpus Preparation: The ROS Wiki extensively documents robotic tasks, actions, and components (see Fig. 3). It serves as the primary data source for constructing the Data Layer. Before processing, the corpus is preprocessed to remove irrelevant content like formatting tags and non-informative text, focusing the analysis on meaningful content.

2) Keyword Recognition and Extraction Process: The Data Layer recognises and extracts keywords from the ROS Wiki Text corpus. These keywords correspond to high-level

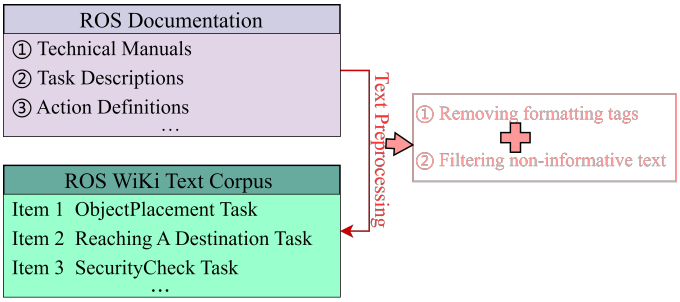

Fig. 3: Text Corpus Preparation

concepts defined in the Pattern Layer and are extracted in a structured format for integration into the Multi-source Heterogeneous Knowledge.

Algorithm 2 Keyword Extraction from ROS Wiki Text Corpus

```
Input: ROS Wiki Text Corpus
Output: Extracted Keywords in Structured Format
keywords ← structure_keywords(filter_keywords (extract_noun_phrases(pos_tag(tokenize(preprocess(corpus))))))
function PREPROCESS(text)
    return remove_special_characters(remove_html_tags(text.lower()))
end function
function TOKENIZE(text)
    return nltk.word_tokenize(text)
end function
function POS_TAG(tokens)
    return nltk.pos_tag(tokens)
end function
function EXTRACT_NOUN_PHRASES(tags)
    return extract_from_tree(nltk.RegexpParser("NP: {<DT>?<JJ>*<NN>}").parse(tags))
end function
function FILTER_KEYWORDS(phrases)
    return [p for p in phrases if is_relevant(p)]
end function
function STRUCTURE_KEYWORDS(keywords)
    return structure_into_table(keywords)
end function
return keywords
```

As illustrated in Algorithm 2, the keyword extraction process begins with preprocessing the ROS Wiki text, followed by tokenization and POS tagging to identify relevant noun phrases. These phrases are then filtered and structured into a tabular format, as shown in Table I. This structured format categorizes extracted terms into task names, descriptions, actions, preconditions, and postconditions.

3) Integration with Pattern Layer: Once the relevant keywords have been extracted from the ROS Wiki Text corpus, they are mapped onto the high-level concepts defined in the Pattern Layer. This mapping process involves aligning the extracted keywords with the corresponding nodes and relationships in the Multi-source Heterogeneous Knowledge.

4) Data Layer Population: The final step in building the Data Layer involves adding the extracted and mapped keywords to the Multi-source Heterogeneous Knowledge. Each keyword is integrated as a node or attribute, enriching the graph with ROS Wiki Text corpus data. This ensures the knowledge graph reflects real-world robotic tasks and enhances practical application.

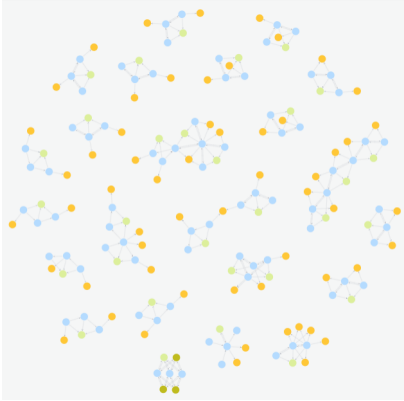

Fig. 4: The Multi-source Heterogeneous Knowledg Graph

We create a scalable and reusable knowledge graph that converts unstructured text data into structured representations by integrating this data with the Pattern Layer. As shown in Fig. 4, the graph contains 210 nodes across 22 types and 288 relationships.

IV. Trustworthy Robot Behavior Tree Generation

A. Overview

This chapter details the generation of robot plans using Multi-source Heterogeneous knowledge and behavior tree (BT) architectures, focusing on symbolic plan generation through knowledge graph reasoning and implementing these plans using BTs. Fig. 5 shows the process of trustworthy robot behavior tree generation.

B. Symbolic Plan Generation by Knowledge Graph Reasoning

Symbolic plan generation via knowledge graph reasoning transforms high-level tasks into practical robot sequences. Python is employed to query the knowledge graph and merge action pairs into a preliminary robot plan.

1) Partial Action Pairs by Knowledge Graph Reasoning: Python scripts query the Multi-source Heterogeneous Knowledge in Neo4j to extract action pairs representing sequential actions for specific tasks. Task names are obtained through keyword extraction (Section 3.3) as query inputs.

Algorithm 3 shows how Neo4j is queried for action pairs:

C. Robot Behavior Tree Generation

This section presents the implementation of robot plans using behavior trees (BTs), offering a modular and hierarchical structure for robot actions. It covers two key aspects:

TABLE I: The Structured Format of Extracted Keywords

Task	Description	Related Actions	Preconditions
Object Placement	Place an object in the designated location	PickUpObject, MoveToLocation, PlaceObject	ObjectDetected, ObjectGrasped
Reaching A Destination	Navigate to the specified destination	MoveForward, AvoidObstacle	DestinationSet

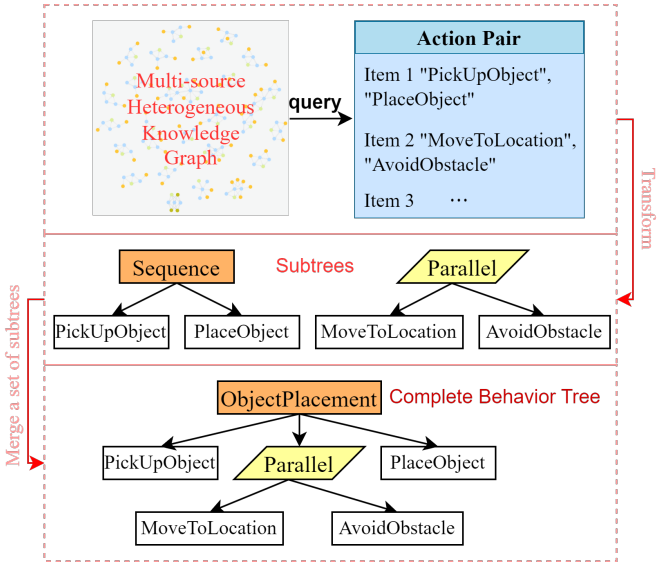

Fig. 5: The Process of Trustworthy Robot Behavior Tree Generation

Algorithm 3 Query Knowledge Graph for Action Pairs

Input: Task name T, Neo4j knowledge graph
Output: List of action pairs
Connect to Neo4j
query ← **"MATCH (a:Action)-[:PRECEDES]-¿(b:Action) WHERE a.task = T RETURN a.name, b.name"**
results ← Execute query
action_pairs ← []
for all result in results **do**
    action_pairs.append((result['a.name'], result['b.name']))
end for
return action_pairs

transforming high-level plans into BT structures and merging local BTs into a complete executable plan.

1) Transform Action Pairs to Behavior Tree Subtrees: The next step involves transforming action pairs from knowledge graph reasoning into behavior tree subtrees. The transformation follows these rules:

- **Action Pair to Sequence Node:** A sequence node is created for each action pair, with the first action as the first child and the second action as the second child.
- **Standalone Actions as Leaf Nodes:** Actions without preceding or following actions are treated as standalone leaf nodes.

This process converts action pairs into subtrees, where each subtree represents a sequence of actions executed in a specific order.

2) Merge a Set of Subtrees into Complete Behavior Tree: After generating the behavior tree subtrees, the final step is to merge them into a single behavior tree representing the complete robot plan. The merging follows specific rules, as outlined in Algorithm 4:

Algorithm 4 Merge Behavior Tree Subtrees

Input: List of subtrees
Output: Complete behavior tree
root ← CreateRootNode()
control_node ← CreateSequenceNode() ▷ Or Selector Node
AddChild(root, control_node)
for all subtree in subtrees **do**
    if RequiresParallelExecution(subtree) **then**
        parallel_node ← CreateParallelNode()
        AddChild(parallel_node, subtree)
        AddChild(control_node, parallel_node)
    else
        AddChild(control_node, subtree)
    end if
end for
return root

Merging Rules:

- Root Node: A sequence or selector node governs the main control flow.
- Subtree Merging: Sequence nodes handle ordered tasks, and selector nodes manage alternative actions.
- Parallel Actions: For parallel execution, subtrees are grouped under a parallel node.

These two algorithms transform action pairs into behavior tree subtrees, which are then merged into a complete tree structure according to specific rules. This achieves a visual representation of the robot plan and ensures its logical consistency and executability.

V. Experiments

A. Experimental Setup

We conducted experiments to validate the effectiveness and efficiency of our approach in generating clear, accurate, and efficient behavior tree architectures for specific tasks. The baseline approaches are selected as follows:

- **GPT4-based approach**: we utilize the latest GPT4 model as the behavior tree generator that receives specific task descriptions as inputs and directly outputs the detailed plan steps to complete the tasks. The logical accuracy and time costs of output plan steps are measured in the experiment.

- **Component-based approach**: we utilize the most popular BehaviorTree.CPP[3] component-based development framework to manually design the behavior tree structure for specific task goals, as well as the existing ROS components for behavior tree node implementation.

In the experiment, we quantitatively evaluate the performances of task plan outputs and behavior tree structure generation by following metrics:

- **Task plan accuracy**: The correctness of the task plan steps that generated by each approach for achieving the task goals.
- **Task plan consistency**: The consistency of generated task plans for the same task goal when experimenting with an approach multiple times.
- **Behavior tree generation time**: The overall time costs of generating the plan steps and transforming them into a complete and practical behavior tree structure.

The experiment has designed a list of 100 common robotic tasks for repetitive trials to obtain the statistics as objectively as possible, which covers a wide range of robotic types and task goals.

B. Results and Analysis

Fig. 6 shows the comparison of different Methods. Table II presents a summary of the evaluation metrics for each method.

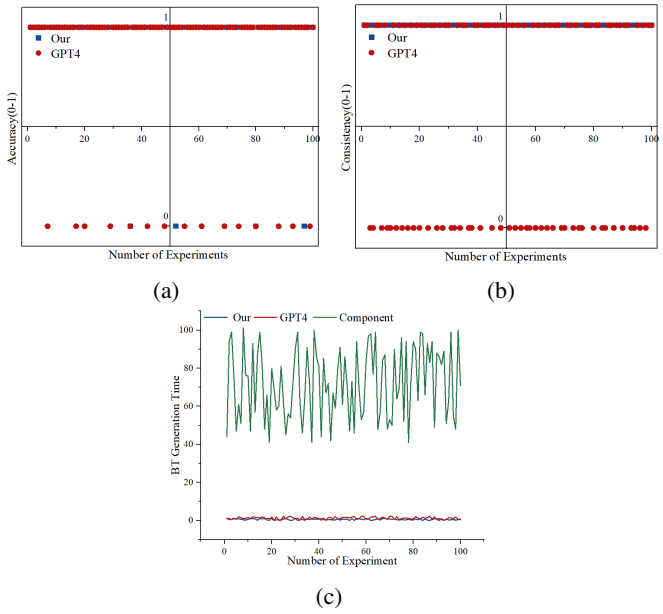

Fig. 6: Comparison of different Methods: Figures a and b show the comparison of our method with GPT 4 in terms of accuracy and consistency, respectively. Figure c shows the comparison of our method with GPT 4 and component-based methods in terms of behavior tree generation time

[3]https://github.com/BehaviorTree/BehaviorTree.CPP

TABLE II: Comparison of different Methods

Methods	Accuracy	Consistency	BT Generation Time
Our	96%	100%	0.62s
GPT4	85%	56%	1.23s
Component	-	-	72s

1) Task plan accuracy: Fig. 6a compares the task plan accuracy between our proposed method (blue) and GPT-4 (red). Our method maintains a high accuracy rate of 96%, consistently generating correct task plans (value = 1) across almost all experiments. GPT-4, on the other hand, achieves an accuracy of 85%, with a notable number of incorrect plans (value = 0) appearing throughout the trials.

2) Task plan consistency: Fig. 6b illustrates the task plan consistency of our proposed method (blue) compared to GPT-4 (red). Consistency is measured by whether the generated task plans remain the same across multiple experiments with the same input. Our approach maintains perfect consistency (value = 1) in all experiments, ensuring the generated task plans are identical and reliable. GPT-4, on the other hand, demonstrates a significant lack of consistency, with multiple variations in task plans (value = 0) across many trials.

3) Behavior tree generation time: Fig. 6c shows the BT generation times for our method, GPT-4, and a component-based approach. Our method (blue) is the fastest, averaging 0.62 seconds per experiment, while GPT-4 (red) averages 1.23 seconds with minor fluctuations. The component-based approach (green) is much slower, varying between 50-100 seconds, reflecting the inefficiency of manual generation compared to automated methods.

The experimental results indicate that the Multi-source Heterogeneous Knowledge (MHKG)-driven approach outperforms the GPT-4 and ROS-based component approaches across several key metrics. The MHKG-driven method offers high task plan accuracy, high task plan consistent performance and fast behaviour tree generation time, making it a superior choice for designing robotic software architectures.

VI. Conclusion

In this paper, we proposed a novel approach for generating trustworthy behavior trees (BTs) using a multi-source heterogeneous knowledge graph. Compared to existing approaches, the method improves the accuracy, consistency, and time of BT generation. Experimental results confirm our method's high trustworthiness and reduced time costs, making it a scalable solution for dynamic robotic tasks.

We plan to refine the knowledge graph for future work by incorporating real-time data sources and enhancing the reasoning mechanisms with advanced AI techniques, aiming for broader deployment in various robotic applications.

References

[1] M. Colledanchise and P. Ogren, "Behavior Trees in Robotics and AI: An Introduction," *CRC Press*, 2017.
[2] P. Ogren, "Modular and Hierarchical Control Systems for Robotics," *Robotics Research*, Springer, 2018, pp. 169-185.

[3] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ogren, "Towards a Unified Theory of Behavior Trees," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 662-667.

[4] J. A. Bagnell, J. A., M. Colledanchise, P. Ogren, and A. Stentz, "Behavior Tree-Based Task Planning in Robotics," *Robotics Research*, Springer, 2019, pp. 255-272.

[5] V. Iovino, F. Pecora, and L. Karlsson, "Designing Behavior Trees for Complex Robotics Tasks," *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1049-1055.

[6] M. Colledanchise and P. Ogren, "Automatic Generation of Behavior Trees from Task Planning Models," *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3729-3734.

[7] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2718-2725.

[8] J. Á. Segura-Muros and J. Fernández-Olivares, "Integration of an automated hierarchical task planner in ROS using behavior trees," *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, IEEE, 2017, pp. 20-25.

[9] A. Joon and W. Kowalczyk, "Leader–follower approach for non-holonomic mobile robots based on extended Kalman filter sensor data fusion and extended on-board camera perception controlled with behavior tree," *Sensors*, vol. 23, no. 21, pp. 8886, 2023.

[10] T. Ribeaud and C. Zhang Sprenger, "Behavior trees based flexible task planner built on ROS2 framework," *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022, pp. 1-4.

[11] S. Yang, X. Mao, Y. Lu, and Y. Xu, "Towards a behavior tree-based robotic software architecture with adjoint observation schemes for robotic software development," *Automated Software Engineering*, vol. 29, no. 1, pp. 31, 2022.

[12] Z. Han, *Robot Explanations: Preferences, Generation, and Communication*, Ph.D. dissertation, University of Massachusetts Lowell, 2021.

[13] M. G. Arenas, T. ao, S. Singh, V. Jain, A. Ren, Q. Vuong, J. Varley, et al., "How to prompt your robot: A promptbook for manipulation skills with code as policies," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 4340-4348.

[14] L. Sun, D. K. Jha, C. Hori, S. Jain, R. Corcodel, *Zhu*, M. Tomizuka, and D. Romeres, "Interactive planning using large language models for partially observable robotic tasks," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 14054-14061.

[15] G. Tziafas and H. Kasaei, "Lifelong robot library learning: Bootstrapping composable and generalizable skills for embodied control with language models," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 515-522.

[16] H. Zhou, Y. Lin, L. Yan, J. Zhu, and H. Min, "LLM-BT: Performing robotic adaptive tasks based on large language models and behavior trees," *arXiv preprint arXiv:2404.05134*, 2024.

[17] Y. Chen, J. Arkin, Y. Zhang, N. Roy, and C. Fan, "Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 4311-4317.

[18] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "ISR-LLM: Iterative self-refined large language model for long-horizon sequential task planning," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 2081-2088.

[19] W. Zu, W. Song, R. Chen, Z. Guo, F. Sun, Z. Tian, W. Pan, and J. Wang, "Language and sketching: An LLM-driven interactive multimodal multitask robot navigation framework," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 1019-1025.

[20] E. Bartoli, F. Argenziano, V. Suriani, and D. Nardi, "Knowledge acquisition and completion for long-term human-robot interactions using knowledge graph embedding," *International Conference of the Italian Association for Artificial Intelligence*, Cham: Springer International Publishing, 2022, pp. 241-253.

[21] G. Wilcock and K. Jokinen, "Conversational AI and knowledge graphs for social robot interaction," *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2022, pp. 1090-1094.

[22] Y. Zhou, S. Zhu, W. Song, J. Gu, J. Ren, X. Xi, T. Jin, and Z. Mu, "Robot planning based on behavior tree and knowledge graph," *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2022, pp. 827-832.

[23] Y. Song, A. Li, H. Tu, K. Chen, and C. Li, "A novel encoder-decoder knowledge graph completion model for robot brain," *Frontiers in Neurorobotics*, vol. 15, pp. 674428, 2021.

[24] N. Xu, W. Wang, R. Yang, M. Qin, Z. Lin, W. Song, C. Zhang, J. Gu, and C. Li, "Aligning knowledge graph with visual perception for object-goal navigation," *arXiv preprint arXiv:2402.18892*, 2024.

[25] Y. Zhao, X. Mao, and Y. Yang, "An effective method for constructing a robot operating system node knowledge graph based on open-source robotics repositories," *Electronics*, vol. 12, no. 19, pp. 4022, 2023.

[26] R. Miao, Q. Jia, F. Sun, G. Chen, H. Huang, and S. Miao, "Semantic representation of robot manipulation with knowledge graph," *Entropy*, vol. 25, no. 4, pp. 657, 2023.

[27] A. Daruna, D. Das, and S. Chernova, "Explainable knowledge graph embedding: Inference reconciliation for knowledge inferences supporting robot actions," *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 1008-1015.

[28] J. Deng, T. Wang, Z. Wang, J. Zhou, and L. Cheng, "Research on event logic knowledge graph construction method of robot transmission system fault diagnosis," *IEEE Access*, vol. 10, pp. 17656-17673, 2022.

[29] G. Wilcock and K. Jokinen, "Should robots indicate the trustworthiness of information from knowledge graphs?," *2022 10th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, IEEE, 2022, pp. 1-3.

[30] M. Beetz, D. Bessler, A. Haidu, et al., "KnowRob 2.0: A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 512-519.

[31] J. Bateman, et al., "Heterogeneous ontologies and hybrid reasoning for service robotics: The EASE framework," *ROBOT 2017: Third Iberian Robotics Conference: Volume 1*, Springer International Publishing, 2018, pp. 417-428.