# Knowledge-based Execution Configuration of Behavior Trees*

Oriol Ruiz-Celada[0000−0002−6171−7270], Jan Rosell[0000−0003−4854−2370], and Raúl Suárez[0000−0002−3853−7095]

Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya
Barcelona, Spain
{oriol.ruiz.celada,jan.rosell,raul.suarez}@upc.edu

**Abstract.** Automated planning is commonly used to obtain plans to solve particular tasks. To execute the plans, Behavior Trees (BTs) have emerged as a popular execution architecture due to their reactivity and modularity. Configuring the execution of a plan into a BT requires expanding high level actions into the proper BT structure. For this, the use of pre-defined rigid templates is common. In this work, in order to overcome the limitation of these methods, we present a new approach to describe robotic behaviors using ontologies and to adapt BTs to current specifications of the task and the world by reasoning on what modifiers to their base template need to be applied. These modifiers and their properties are formally defined using ontologies. We present a proof of concept of these modifiers applied to a base BT of a Pick operation.

**Keywords:** Execution Configuration · Ontologies · Behavior Trees.

## 1 Introduction

In this work, we address the challenge of enabling robots to adapt their actions to dynamic and unstructured environments. While off-the-shelf planners can generate correct action sequences to solve particular problems, integrating acting with planning remains of great interest. We refer to Execution Configuration as a module that translates a high-level task plan into an execution structure that the robot can perform.

Previous approaches to execution configuration have relied on either implicit representations or fixed templates for each planned action, limiting adaptability [1] [2] [3]. In contrast, we propose a method that offers greater flexibility in how robotic behaviors can be represented, tailoring the execution to the specific tasks and environment. Behavior Trees (BTs) offer a natural way of representing robot behaviors. Control nodes regulate the execution flow by combining

* This work was supported by the European Commission's Horizon Europe Framework Programme with the project AI-Powered Manipulation System for Advanced Robotic Service, Manufacturing and Prosthetics (IntelliMan) under Grant Agreement 101070136

the outcomes of child nodes, while leaf nodes represent primitive actions. The modularity of BTs make them a suitable structure for the approach presented in this work.

To represent domain knowledge, we employ ontologies, facilitating logical reasoning and inference. We integrate the Planning Domain Definition Language (PDDL) and BTs through ontology-based descriptions of robot behaviors, enabling richer representations and adaptability in execution configuration than PDDL and BTs can represent on their own. The benefit of utilizing a knowledge-based methodology allows for versatility and being able to work across various domains, significantly expediting the development of new robot functionalities [4]. The contributions of this work are:

- Novel characterization of domain and execution knowledge to define robot behaviors integrating PDDL, BTs and ontological knowledge.
- Enhanced ontology-based description of robot behaviors, enabling a range of variations of BT structures and avoiding relying only on predefined templates.
- Knowledge-driven dynamic generation of adaptive BTs for task plan Execution Configuration, with reasoned incorporation of non-functional actions alongside planned execution to include adaptive capabilities.

## 2 Framework Description

This paper contributes to the development of Be-Aware, an ontology-based adaptive robotic manipulation framework [5]. Be-Aware is designed for dynamic environments and utilizes Knowledge Representation and Reasoning to enable awareness of the surroundings. The framework allows planning actions based on this awareness, monitoring their execution, and adapting to disruptions. Be-Aware uses the proposed Execution Configuration module to generate BTs adapted to the current scenario. These trees include monitoring and recovery branches to facilitate adaptation to unexpected changes during execution.

The process to obtain the BT to execute to achieve a goal, called Output BT, is illustrated in Figure 1. The Planning Design module automatically generates the PDDL files to solve the task using the knowledge in the Knowledge Base. An off-the-shelf PDDL planner is used to obtain a high-level Task Plan, e.g. `pick(rob1,can1,table1)`→`place(rob1,can1,table2)`. The Execution Configuration module, detailed in the next section, acts as a bridge between task planning and execution by producing the Output BT, relying on the Knowledge Base to adapt BT templates to the current situation. The Output BT is executed by a BT executor. This process can be repeated during execution, modifying the PDDL files and generating a new Output BT in cases where disturbances require replanning the current execution.

The Knowledge Base Manager, illustrated in Figure 1, governs the Knowledge Base, essential for the framework's functionality. It contains two types of knowledge: Ontological Knowledge and Other Knowledge. A framework-specific Be-Aware Ontology contains the definitions of the concepts required for Execution Configuration and the other modules in the framework. The user has to

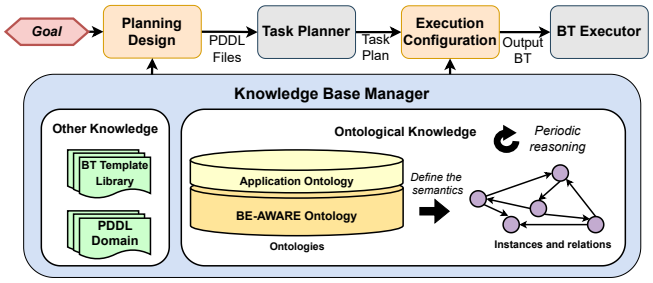



Fig. 1: Top, Pipeline to obtain a configured Output BT using the Knowledge Base. Bottom, Knowledge Base Manager and its knowledge.

define their own Application Ontology for the concepts and relations specific to their scenario, allowing for different sources of knowledge tailored to the specific scenario to be used for reasoning. A reasoner is used to continuously infer new facts and relations. The Other Knowledge contains non-ontological data, like the PDDL files for planning and the BT Template Library.

## 3 Execution Configuration Module

The Execution Configuration introduced in this work leverages semantic information and reasoning to dynamically configure BT structures based on current knowledge. For example, when executing a Pick action, the system may need to assess whether an object can be grasped in its current position or needs to be repositioned first. Numerous adjustments can be made to a Base Template of a BT to better fit a specific problem instance, but the modifiers must be formally specified and implemented. These modifiers are named "Flavors", and are formed by the following components:

- **FlavorTargets:** Parts of the Base Template affected by the Flavor, including branches, nodes or parameter values.
- **Trigger:** Condition that determines whether the Flavor must be applied. For instance, a Flavor that modifies the execution velocity may be applied if the picked object belongs to the `DangerousObject` subclass. They are checked through SPARQL queries to the Knowledge Base.
- **Keyword:** Type of the Flavor. Examples include `IgnoreA` for eliminating behavior A, and `DelayAUntilAfterB` for sequencing A after B.

The Flavor approach allows adding flexibility and generality to the description of the behaviors that the robot can perform. Moreover, flavors can be used to enable reasoned integration of monitoring and replanning branches into the execution BT.

The Execution Configuration process is illustrated in Figure 2. It begins by parsing each part of the Task Plan into a Seed, a structure formed by the behavior

to be configured, (such as `pick`) and the parameters for the current plan (such as `rob1, can1, table1`). The corresponding BT Base template is loaded. Then, the flavors of the behavior are applied, modifying the final structure of the BT. The parameters in the seed are instances of concepts in the Knowledge Base, with a series of properties, and thus allowing the system to check if the flavor triggers are active or not. After all the flavors have been applied, there may be remaining Subtree calls that need to be expanded. New seeds are parsed, and this process is repeated until all the leaves in the BT are BT Action Nodes.

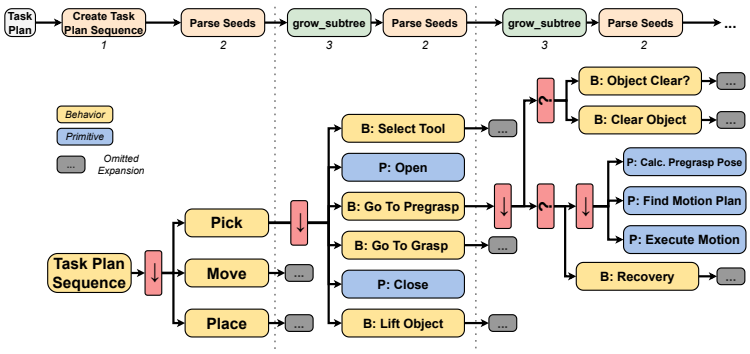


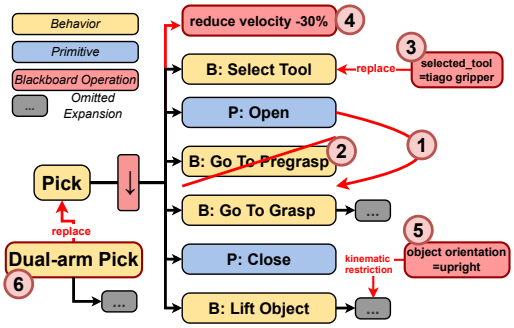

Fig. 2: Execution Configuration pipeline (above), accompanied by an example (below) of a configured Output BT (the BT has been simplified and not fully expanded due to space constraints).

A library of Flavors has been developed to serve as a proof of concept and illustrates the variety of possibilities of the proposed approach, and it can be extended. Figure 3 shows an example of some of the flavors implemented modifying the template of the `pick` behavior. The approach allows for new Flavor keywords to be added with relative ease, expanding the adaptation possibilities.

## 4 Conclusions and Future Work

Through this proposed method, it becomes possible to configure the execution of plans encompassing manipulation, perception, navigation, monitoring, and recovery within a BT. This adaptation to different scenarios is achieved through reasoning and the application of Flavors to base templates. Our approach offers the following advantages over existing ones:

- **Consistency:** Flavors ensure consistent application of modifications to BT base templates.
- **Scalability:** Flavors allow for scalability in the amount of modifications applied to BT templates. Designing scenarios for modifications can become cumbersome with traditional methods, but flavors enable quicker and easier creation and assignment of modifications.




| Flavor Description |
|---|
| 1.If the robot has a hand, **Delay** `Open` until after `GoToPregrasp` |
| 2.If the object is small, **Ignore** `GoToPregrasp` |
| 3.If the robot only has one possible tool, **Replace** `SelectTool` by directly setting the tool in the BT |
| 4.If the object is dangerous, **Reduce** velocity by 30% |
| 5.If the object is a glass with liquid, add the **Kinematic restriction** to be kept upright |
| 6.If the object is heavy, **Replace** `Pick` by `Dual-arm Pick` |

Fig. 3: Flavors modifying an original `Pick` template, indicating the modifications in red. The flavors would only be applied in case that their triggers are true.

- **Generality:** The approach enhances generality in behavior descriptions. Basic BT templates can be reused across different heterogeneous domains, with adaptations made only through the design of different assortments of flavors.

Further research will focus on expanding available flavor keywords and enhancing their application, particularly in how multiple flavors interact. There is also potential in using optimizing algorithms after applying the flavors to refine the BT further. Additionally, the development of behaviors and flavors for monitoring and recovery is planned, integrating them into BTs for robust task execution. A current limitation is the reliance on available knowledge for trigger checks, which may only be known during behavior execution. Future efforts will explore reasoning techniques to detect triggers needing verification within the BT itself and how flavors can be utilized in such cases. This work is currently being implement to be validated in simulation and real-world scenarios.

## References

1. Ruiz-Celada, O., Verma, P., Diab, M., Rosell, J.: Automating adaptive execution behaviors for robot manipulation. IEEE Access, vol. 10, pp. 123 489–123 497, 2022.
2. Rovida, F. et al.: SkiROS—A skill-based robot control platform on top of ROS. Studies in Computational Intelligence. pp. 121-160 (2017). 10.1007/978-3-319-54927-9_4.
3. Beetz, M., Kazhoyan, G., Vernon, D., Member, S.: The CRAM Cognitive Architecture for Robot Manipulation in Everyday Activities. April 2023. [Online]. Available: https://arxiv.org/abs/2304.14119v1
4. Olivares-Alarcos, A. et al.: A review and comparison of ontology-based approaches to robot autonomy. The Knowledge Engineering Review, vol. 34, 2019.
5. Ruiz-Celada, O., Dalmases, A., Suárez, R., Rosell, J.: BE-AWARE: an ontology-based adaptive robotic manipulation framework. IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–4 (2023)