Policy-Enhanced Fallback Nodes in Behavior Trees

Andreas Naoum[1], and Loizos Michael[1,2]

Abstract— **Robots operating in highly-dynamic environments are burdened with the task of continuously monitoring and flexibly adapting to their changing surroundings and internal state. Fallback nodes in Behavior Trees offer a basic means to achieve this adaptability, by determining a total static ordering over contingency plans for achieving a goal. Attempting to encode a more nuanced contextual ordering leads to significantly larger Behavior Trees, sacrificing their human readability and the ease of their elaboration. Towards maintaining these two desiderata, we propose the enhancement of fallback nodes with symbolic policies that transparently determine, at execution time, the order in which contingency plans are to be considered according to the current context. Based on our undertaken user study, we offer evidence for the benefit of employing such policy-based Behavior Trees over alternative versions of Behavior Trees.**

I. Introduction

Behaviour Trees (BTs) are becoming popular in robotics because they offer several benefits for handling dynamic situations and defining reactive behaviours [1], [2]. Researchers have been focusing on making BTs adaptable to uncertain environments, highlighting the potential of robotics in diverse real-world domains [2]. While robots can often operate autonomously, there are cases where it is important for their behavior to be transparent to humans, and even allow them to intervene. In this context, transparency and explainability are crucial in developing human-centred robots.

One potential benefit of improving BTs is the increased ability to effectively address failures in autonomous agents. Despite significant attempts to enhance the reliability of autonomous agents, these agents face frequent failures when operating in dynamic environments. Such failures can pose risks to the environment, the autonomous agent itself, and even human safety [3]. Dealing with failures or unexpected events in BTs can be achieved by incorporating safety guarantees and recovery plans into the task. Many approaches have been developed to guide the agents and resolve failures in such scenarios with or without human involvement [4], [5]. However, challenges arise with hard-coded, large and complex sub-trees for safety and recovery. This emphasizes the necessity to improve the adaptability and transparency of BTs, and enabled them to be modified by an external entity.

Building on the need for transparency and explainability, fallback nodes in BTs often represent critical decision points, and the capability to communicate the decision-making process is essential. Moreover, enabling external entities to coach robots can address misalignment in Human-Robot Interaction (HRI). This makes interactive machine learning an appropriate approach for determining the optimal actions to execute in fallback nodes. Machine Coaching [6], a form of interactive machine learning, aims to develop AI systems that are not only explainable but also allow human supervision. As the author mentioned, the decision-making for critical tasks can be retained, compared to machine learning approaches where this is not an option as these solutions are based on statistical solutions. The proposed two-step approach involves acquiring world knowledge through machine learning and then applying machine coaching to avoid undesirable inferences, thereby allowing the system to incorporate user-specific knowledge while reducing bias from the training data. Building such an integrated system poses challenges, primarily in developing a seamless interface that enables humans to interact with and modify the decision-making process without extensive technical knowledge.

In addressing these challenges, our research introduces a novel approach to BT structure aimed at improving key aspects of HRI. We hypothesize that our proposed approach will lead to improvements in three main areas: transparency, adaptability, and trust. Specifically, we anticipate that participants will find the decision-making with our approach to be more transparent and easier to understand compared to other extensions of BTs. Furthermore, we predict that our approach will be perceived as user-friendly, making it simple for users to modify and extend the behavior of the robot. Lastly, we expect that users will exhibit higher levels of trust and confidence in the robot's decisions when using our proposed BT framework, compared to the other approaches.

II. Related Work

Various extensions have been suggested, emphasizing fallback implementation [1], [2]. Initially, approaches suggested by researchers involve the incorporation of a utility system into the BT structure. Merrill [7] introduced the concept of Utility Fallback Node, which integrates a utility calculation function for each node. The utility fallback node then dynamically sorts its children based on their utility values, thereby determining the fallback children nodes order.

More sophisticated frameworks to deal with uncertainty lead to other solutions. Stochastic BTs [8] also use a utility system and rely on the acquired experience, as they require training and taking into account the success probabilities associated with each child node based on Hidden Markov Models (HMM). These models are utilized to handle noisy observations and estimate state transition probabilities within the BT. By running Monte Carlo simulations, the authors verified analytical results about Stochastic BTs for complex robotic tasks. Additionally, a hybrid combination of active

[1]CYENS Centre of Excellence, Nicosia, Cyprus a.naoum@cyens.org.cy

[2] Open University of Cyprus, Nicosia, Cyprus loizos@ouc.ac.cy

inference and BTs has been suggested in dealing with uncertainty and reduce the number of nodes [9]. They introduced a new type of leaf node that specifies the desired state, while the action planning is carried out using active inference. Their findings demonstrate an improved runtime adaptability. These frameworks illustrate that probabilistic solutions could be advantageous for complex robotic tasks.

The need to influence human behavior without causing unintended consequences has led to proposed solutions that allow for human guidance [1]. Two concepts, styles [10] and hints [11], have been suggested to temporarily modify the priorities of the BTs. The concept of styles involves the selective disabling of a subset of the BT, offering a flexible approach to modify the tree's behavior temporarily. On the other hand, the hints concept focuses on the reorganization of sub-trees based on external cues. By incorporating hints from an external entity, this approach enables adaptive adjustments to the BT, enhancing its responsiveness to changing conditions and requirements. These solutions may be advantageous for integrating human guidance into robotic systems.

In addition to BT extensions, it is essential for seamless HRI that humans have a clear understanding of the robot's behavior, including its intentions, objectives, capabilities, and decision-making process. Olivia et al. [12] proposed that one effective way to communicate robot policies is by demonstrating examples of robot behaviors. They conducted a user study in which they presented examples of robot behaviors in various situations to end users, categorized as exploratory and critical states. The results indicated that end-users were able to comprehend the robot's policies, assess its trustworthiness, and consequently establish trust in the robot if it was considered trustworthy.

Our proposed solution aims to achieve a balance between these two directions, the necessary runtime adaptability based on the environment and the adaptability required by incorporating human guidelines and preferences, while providing a mental model for the robot's decision-making process.

III. Background

A. Background on BTs

BTs are used to design and execute complex behaviors for autonomous systems. BTs are easy to visually comprehend and simple to design, and they offer modularity, scalability, and reusability. A BT consists of key components, such as control flow nodes, Sequence, Fallback), with the possibility of having children, and execution nodes, Condition and Action. The root of the BT initiates the execution process by periodically sending signals to its children. When a node in the BT is activated, it reports back to the parent with a status of running if its execution is ongoing, success if it has attained its objective, or failure otherwise [1].

Focusing on the fallback node, it is important to highlight some observations based on commonly used design practices. When condition nodes are integrated among the children of the fallback node, the action nodes essentially represent alternative actions that yield the same outcome [1], [9]. In a classical BT, the determination of the next child is based on the design choices made. However, this structure has a significant limitation, as Merill has highlighted [7], the fixed priorities within the children of fallback nodes. Notably, the optimal action may differ from the predetermined design depending on the circumstances and state of the environment. Addressing this constraint requires prompting a shift towards a paradigm that accommodates dynamic considerations within the BT structure.

B. Background on Prudens

Prudens [13] is a declarative programming language, linked to an efficient provable deduction process and is interpreted under an argumentative semantics. As Prudens is based on argumentation, it enables the creation of systems capable of explaining their decisions by providing the internal arguments that led to a conclusion. Moreover, its authors designed the language to be compatible with the knowledge acquisition process of machine coaching, a proven and efficient human-in-the-loop machine learning approach. Prudens empowered cognitive assistants for personalising the user experience as well to promoting user understanding, and therefore, building more trust towards the system. The user-friendly ecosystem of a cognitive assistant demonstrated how machine coaching can be integrated into real-world applications and holds promise for empowering cognitive robots as well.

IV. Methodology

Our strategy revolves around enhancing the fallback node within the BT structure. To empower the agent's capacity for reasoning, we integrate prudens, and give the system a symbolic presentation of its environment and capabilities. Through the context and policy, the agent engages in logical deduction to prioritize child nodes over their siblings. Subsequently, we adapt the fallback node implementation to incorporate this reasoning framework.

A. Reasoning System

In order to provide the system with a symbolic representation of its environment and abilities, we have established the context and the policy. The context consists of a set of fluents that capture observable aspects of the agent's status and environment. For instance, fluents could denote the presence of an object in a room, its attributes, the state of a gripper (whether it is open or closed), or any other dynamic property. The current context serve as the input for decision-making, providing real-time information about the environment and the system's state. The policy is composed of rules that guide the decision-making process of the fallback node.

The reasoning system infers constraints for the ordering, represented as *before(childX, childY)*, which denotes the prioritisation of child X over child Y. Alternatively, for a list of pairs of indeterminate size, the notation *order([(childX, childY), ...])* is used to specify the prioritisation of multiple children over others. This design choice was made for adaptability in dynamic environments where priorities can shift rapidly. It enables the updating of the priority schema with minimal disruptions to the overall system functionality.

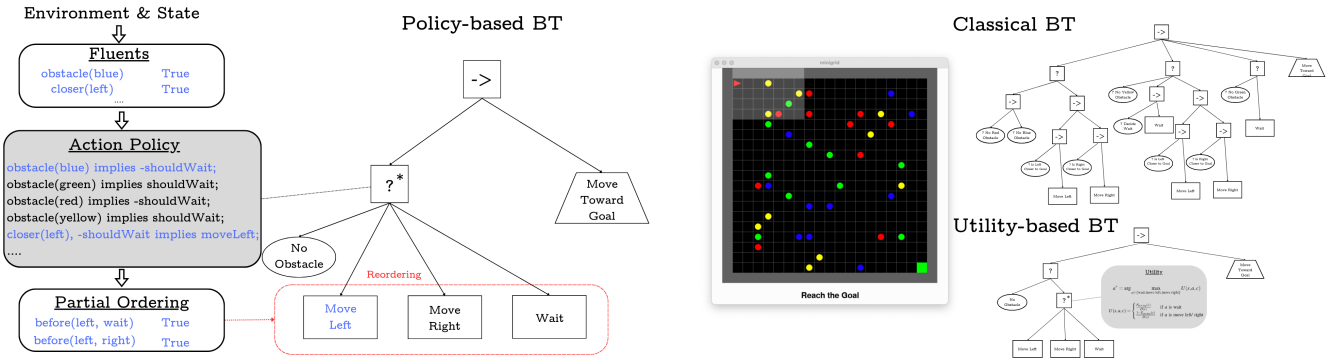

Fig. 1: MiniGrid Environment and Solutions

B. Fallback Node Modification

The current implementation of the fallback node's `Tick` function iterates through its children based on a predefined order. To enhance the existing implementation, we used the Pytrees library [14] to extend the fallback node's tick function. Instead of following the predefined order, the overridden tick function dynamically decides the prioritized order and subsequently ticks the children in the determined order.

This extension allows the fallback node to iterate through its children according to constraints inferred by the policy. The overridden function guarantees that if the inferences include a constraint such as *before(childX, childY)*, then child X will be ordered before child Y. However, the remaining children are not strictly ordered unless specified. It is also possible to enforce multiple constraints for complex and precise control over the order in which children are ticked. For example, consider the constraint *order([(childX, childY), (childX, childZ)])* where it is prioritised the child X over the children Y and Z. Importantly, there is the opportunity to make the order of execution to only change if the fallback node returns either `SUCCESS` or `FAILURE`, ensuring that the order remains constant during the execution of a child in the `RUNNING` state. This design choice can prevent disruption to a child that is currently executing.

C. Integrating User Advice

Our solution not only addressed ordering constraints, but also enabled easy extension and modification. This flexibility allows for adding or removing rules within the policy, enabling external entities to interact with the agent and adjust its behavior. Presently, our proof-of-concept demonstration includes the functionality to add, remove, or modify rules.

By employing natural language for communication, the robot could explain the policy for a specific scenario and provide clear and comprehensible explanations of its decision-making process. Additionally, the robot could be queried about its actions in particular cases, which allows for preemptive adjustments and refinements. Human advice or feedback could then be seamlessly integrated into the system to ensure that the agent's behavior aligns with human expectations. This bidirectional communication channel can lead to a more human-centered approach to robotics.

V. Evaluation

A. Empirical Design

For our experimental setup, we developed a custom Minigrid environment of an empty room with moving obstacles. The obstacles different by their color and every obstacle based on the color has a different probability to move in the next moment. The goal of the agent was to reach the target goal square, marked as green, without colliding with any obstacle and minimize the steps.

TABLE I: Performance of Each Solution

Type	No. Nodes	Avg. Steps
Classical BT	28	36,7
Utility-Based BT	8	37,3
Policy-Based BT	7	37,0

We developed three different solutions based on (1) classical BT, (2) BT with utility-based fallback nodes, and (3) our proposed solution (see Figure 1). We aimed to ensure a fair comparison among the solutions, as we were interested in the design process and the perception of the system builders. To validate this, we assessed the performance of the three solutions based on the average number of steps required to reach the goal in ten randomly generated scenarios in a 20×20 minigrid environment. All three solutions yielded comparable results, making them equally viable, as detailed in Table I.

The first solution was implemented using a classical BT. Due to the limitations of this method in adapting based on the status and environment, we decided to adopt a fixed strategy for the agent's actions. In this strategy, the agent turns when facing blue or red obstacles, randomly chooses to turn or wait when facing yellow obstacles, and waits when facing green obstacles. These decisions were based on the probabilities of movement for each type of obstacle: blue and red obstacles have a low probability of moving, yellow obstacles have a medium probability of moving, and green obstacles have a high probability of moving.

The second solution was implemented using a BT with utility-based fallback nodes. The fallback nodes as safety guarantees of obstacle avoidance in the classical BT have been replaced by one utility-based fallback node.

Let a be the set of possible actions {wait, move left, move right}. We wish to find the action a^* that maximizes the utility function U. This can be formally expressed as:

$$a^* = \arg \max_{a \in \{\text{wait,move left,move right}\}} U(s,a,c)$$

The utility function calculates the utility for each action:

$$U(s,a,c) = \begin{cases} \frac{P_{moving}(c)}{D(s)} & \text{if } a \text{ is wait} \\ \frac{1-P_{moving}(c)}{D(s)} & \text{if } a \text{ is move left/ right} \end{cases}$$

where $P_{moving}(c)$ represents the probability that an obstacle of color c will move in the next moment, and $D(s)$ is the Manhattan distance from the agent's current position to the target goal square.

In this formulation, the utility for the *wait* action is proportional to the probability that the obstacle will move, making waiting more advantageous when obstacles are likely to move. Conversely, the utility for the *move left* and *move right* actions is inversely proportional to the probability that the obstacle will move. This means that moving left or right is more advantageous when obstacles are less likely to move.

TABLE II: Policy for Fallback Node - MiniGrid Agent

	Rule
1	`probableMove(obs_color) implies shouldWait`
2	`-probableMove(obs_color) implies -shouldWait`
3	`closer(left), -shouldWait implies moveLeft`
4	`closer(right), -shouldWait implies moveRight`
5	`shouldWait implies order([(wait, left), (wait, right)])`
6	`moveLeft implies order([(left, wait), (left, right)])`
7	`moveRight implies order([(right, left), (right, wait)])`

The third solution was implemented using the policy-based approach, which takes into account the probability of obstacles moving and the Manhattan distance to the target goal. The rules are structured to ensure that waiting is more advantageous when obstacles are more likely to move, and turning is prioritized based on the shortest distance to the goal. The rules have been formulated as shown in Table II.

TABLE III: Modification of the Policy

	Rule
1	`obstacle(blue) implies shouldWait`
2	`obstacle(red) implies shouldWait`
3	`obstacle(yellow) implies shouldWait`
4	`obstacle(green) implies -shouldWait`

For demonstrating the ability to modify the behavior, we changed the policy by removing the first and second rule from the initial policy, and we add the rules that guide the agent to wait when faced every obstacle expect the green one. The new rules are shown in Table III.

B. Participants and Procedure

We demonstrated the solutions to 8 people and asked them to fill the questionnaire. Our main criterion was that the participants should had prior experience in robotics by either education or experience. We presented each solution with a live demo and follow-up questions. After the demonstration, participants answered comparison questions. Finally, we modified and extended the action policy in the demo of our proposed solution to showcase the potential of interacting with the action policy.

VI. Results

The results of the user study, shown in Figure 2, provide valuable insights into transparency, adaptiveness, and trust for our system, validating our design choices and highlighting areas for further development.

Firstly, a majority of the participants highlighted the critical importance of adaptiveness to the fallback node and the incorporation of user feedback in improving the BT structure. Our questionnaire revealed that our approach is perceived as more transparent and explainable compared to the utility-based approach. Specifically, it is perceived easier to predict behavior, identify unexpected situations, and understand and modify the decision-making process. In the general comparison, our data shows that utility-based and policy-based approaches are considered adaptable. Both are preferred for adaptability to changes in the environment and for handling unexpected situations. All participants believe that the policy-based approach is more adaptable to changes in requirements. Notably, the classical BT was not chosen for any of the questions. Regarding potential user interaction with policy, participants believe that it is important for the robot to communicate the action policy and that even non-experts could potentially assist in the decision-making process. The ability of the agent to explain the action policy and for humans to modify it is seen to positively affect their trust towards the system.

VII. Conclusion and Future Work

In this paper, we have introduced policy-enhanced fallback nodes for BTs. We have presented a framework which modified the current fallback's node implementation, and utilise Prudens for the symbolic presentation and the inference of the partial ordering. The proposed solution was evaluated and compared with the classical and the utility approach. Results have shown that the proposed solution shows promise in be adopted and guides towards a transparent framework for decision-making in BT without sacrificing effectiveness and efficiency.

For future work, we aim to focus on two possible directions. First, we will develop a framework for humans to understand, modify, and extend action policies in HRI scenarios. Second, we will compare trained action policies with probabilistic solutions, such as stochastic BTs, to evaluate the benefits of each approach.

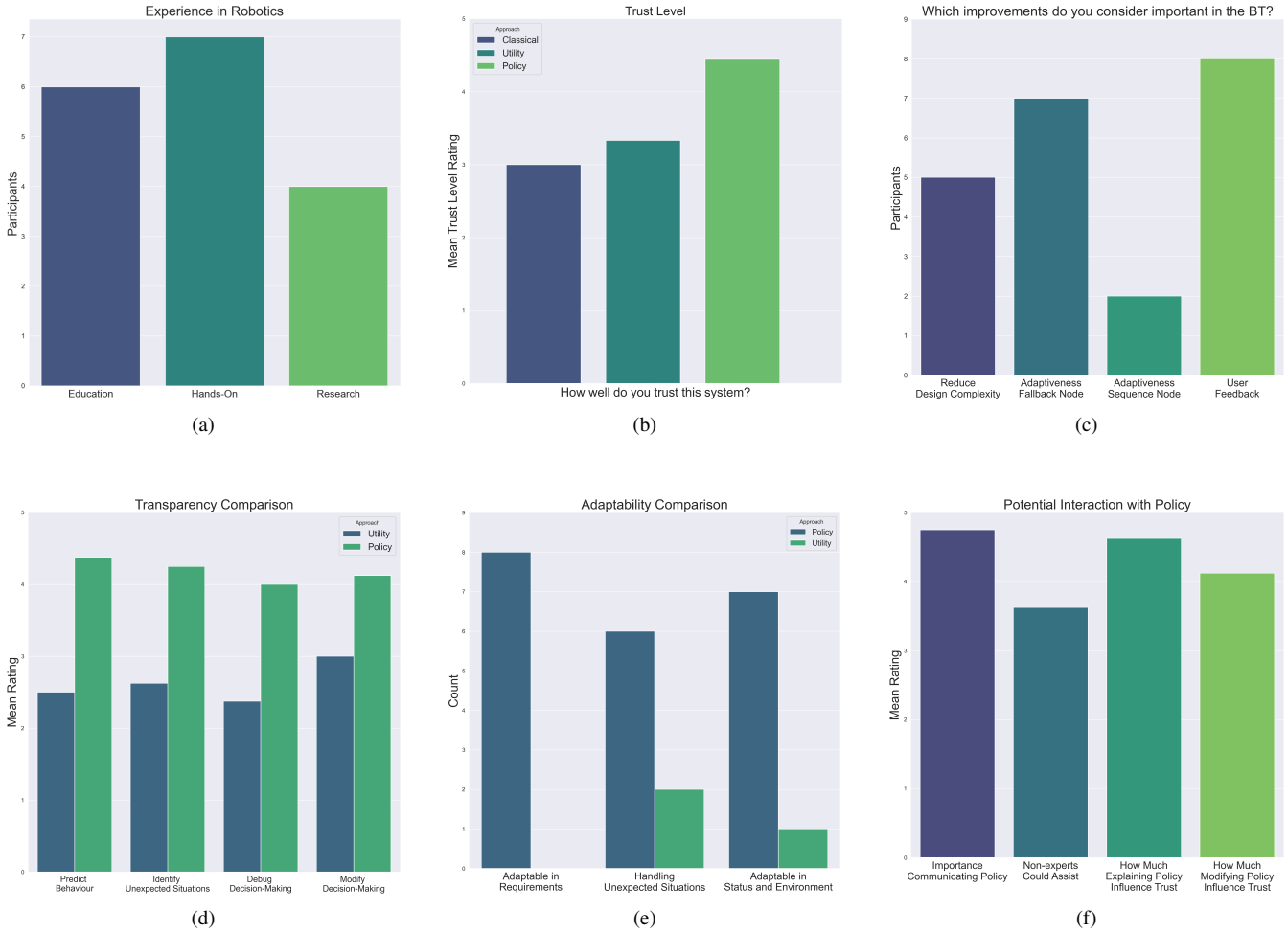

Fig. 2: User Study Results

ACKNOWLEDGMENT

The authors would like to thank Vassilis Vassiliades for his feedback on this work.

This work was supported by funding from the EU's Horizon 2020 Research and Innovation Programme under grant agreement no. 739578, and from the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation, and Digital Policy.

REFERENCES

[1] M. Colledanchise and P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," *CoRR*, vol. abs/1709.00084, 2017.

[2] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A Survey of Behavior Trees in Robotics and AI," *CoRR*, vol. abs/2005.05842, 2020.

[3] B. Dhillon, *Robot system reliability and safety*. Boca Raton, FL: CRC Press, 4 2015.

[4] R. Wu, S. Kortik, and C. H. Santos, "Automated Behavior Tree Error Recovery Framework for Robotic Systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 6898–6904.

[5] S. Honig and T. Oron-Gilad, "Understanding and Resolving Failures in Human-Robot Interaction: Literature Review and Model Development," *Frontiers in Psychology*, vol. 9, 2018.

[6] L. Michael, "Machine Coaching," in *IJCAI 2019 Workshop on Explainable Artificial Intelligence*, vol. Conference Proceedings, Macau, China, 2019, pp. 80–86.

[7] B. Merrill, "Building Utility Decisions into Your Existing Behavior Tree," *Game AI Pro 360*, 2019.

[8] M. Colledanchise, A. Marzinotto, and P. Ögren, "Performance analysis of stochastic behavior trees," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3265–3272.

[9] C. Pezzato, C. H. Corbato, and M. Wisse, "Active Inference and Behavior Trees for Reactive Action Planning and Execution in Robotics," *CoRR*, vol. abs/2011.09756, 2020.

[10] D. Isla, "GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI," Mar 2005.

[11] S. Ocio Barriales, "Adapting AI Behaviors To Players in Driver San Francisco: Hinted-Execution Behavior Trees," 07 2012.

[12] O. Watkins, S. Huang, J. Frost, K. Bhatia, E. Weiner, P. Abbeel, T. Darrell, B. Plummer, K. Saenko, and A. Dragan, "Explaining robot policies," *Applied AI Letters*, vol. 2, no. 4, p. e52, 2021.

[13] V. T. Markos and L. Michael, "Prudens: An argumentation-based language for cognitive assistants," in *Proceedings of the 6th International Joint Conference on Rules and Reasoning (RuleML+RR'22)*. Berlin, Germany: Springer, 12 2022, pp. 296–304.

[14] D. Stonier *et al.*, "Python Implementation of Behaviour Trees." https://github.com/splintered-reality/py_trees, 2024.