Mission planning in Robotic Systems, the PA-BT

Robin Martin

Abstract: *This paper explores various approaches to mission planning for robotic systems, focusing on describing the Planning and Action Behavior Tree (PA-BT) algorithm. It provides a concise overview of an intelligence architecture from Howard et al. (2022) [8] and underscores the pivotal role the Mission Planner plays within this framework. The PA-BT is illustrated through a scenario, followed by an in-depth technical exposition. In addition, the paper briefly explores and compares alternative methods for mission planning towards the conclusion.*

0.0 Introduction:

Stuart Russell emphasized in his paper *The History and Future of AI* (2021) [7] that "AI is concerned principally with… mapping… a stream of raw perceptual data to a stream of actions." In this context, AI's core objective is to bridge the gap between perceiving the environment through sensory data and executing appropriate responses or actions based on that input. This fundamental capability underpins the essence of artificial intelligence, enabling machines to make informed decisions and interact with their surroundings. This understanding of AI's foundational principles serves as a valuable backdrop for exploring the concept of intelligence in robotics.

As we delve deeper into the concept of intelligence in robotics, it becomes evident that robots come in various forms and exhibit different cognitive capabilities. Some common examples include industrial robotic arms used in manufacturing, autonomous vacuum cleaners for household cleaning, and surgical robots employed in minimally invasive medical procedures. It's worth noting that robots don't have to be humanoid or intelligent; they are merely physically situated *agents* that can be programmed to carry out tasks, whether they do so autonomously or not (2019) [6].

The question that naturally arises is: What makes a robot intelligent? According to Murphy (2019) [6], an intelligent robot is one that can perceive its environment and make decisions that "maximize its chances of success." Take, for instance, a traffic light system. While it does possess a level of environmental awareness by sensing the flow of traffic, it falls short of being an intelligent agent. The traffic light system's role is to regulate traffic based on pre-defined rules, switching between "stop" and "go" states. It lacks the ability to make dynamic decisions that would enhance its effectiveness. The lights either turn on or they don't. Since they don't have any ability to optimize their behavior, they are not considered intelligent.

Self-driving cars, by comparison, serve as a prime example of intelligent robots. These physically situated agents employ a combination of sensors, cameras, and artificial intelligence algorithms to not only perceive their surroundings but also make split-second decisions in real time. Whether they're analyzing traffic patterns, detecting pedestrians and other vehicles, or swiftly responding to unpredictable road conditions, self-driving cars demonstrate their ability to operate autonomously within a complex and dynamic environment. The proficiency of self-driving cars in maneuvering with precision, complying with traffic laws, and adjusting to various situations affirms their classification as intelligent robotic systems (2017) [5].

As we explore the capabilities and criteria that define intelligent robots, it becomes clear that their performance hinges not only on their hardware and sensory systems but also on the sophistication of their underlying software. Software empowers robots to not only perceive their environment but also make informed decisions and interact seamlessly with the world around them. Howard et al.'s paper, *An Intelligence Architecture for Grounded Language Communication with Field Robotics* (2022) [8], introduces an intelligence architecture for a robotic system that includes a mission planning algorithm named the Planning and Action Behavior Tree (PA-BT) algorithm. In the following sections, we will delve into the key concepts presented in Howard et al.'s paper, outlining the intelligence architecture and providing a technical explanation of the PA-BT algorithm. This paper aims to shed light on the critical role of software in achieving intelligence in robotics. It illuminates how the Planning and Action Behavior Tree (PA-BT) algorithm contributes to this objective as a central component.

1.0 Intelligence Architecture:

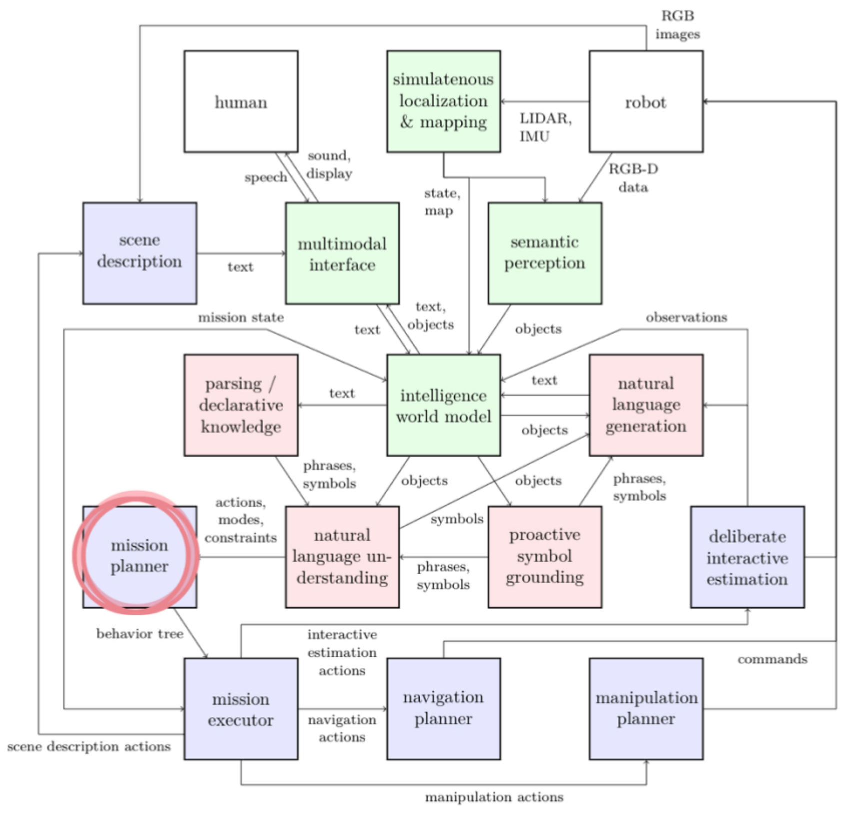


Figure 1 illustrates the intelligence architecture presented by Howard et al. (2022) [8]. Notably, our discussion in this paper centers on the 'mission planner,' highlighted within the diagram, and serves as the central focal point of our discussion.

The architecture in Figure 1 is a system for human-robot collaboration. It consists of different parts that help robots understand and interact with their environment and humans. The main components are:

1. **Perception and Environment Modeling:** This helps the robot make sense of its surroundings.
2. **Grounded Language Communication:** The robot uses this to understand human language and communicate effectively.

3. **Mission Planning and Execution:** This part helps the robot plan and carry out tasks given by humans.

The intelligence architecture functions as the operational backbone of the system, much like an operating system in a computer, which manages the flow of information between software and hardware. Similarly, the intelligence architecture orchestrates the seamless flow of data and instructions between the robot's sensors, processing capabilities, and external communication channels. This orchestration empowers the robot to navigate and respond to its environment autonomously and instantaneously.

The system operates on the Robot Operating System (ROS) to facilitate communication and information exchange between components. At its core, the system relies on an "intelligence world model" as the repository of the robot's knowledge. This model contains rich semantic information about objects and their locations, continuously updating this knowledge base by integrating sensor data and interactions with human users. The synergy between the intelligence architecture and the ROS framework creates a robust foundation for the system's functionality and adaptability.

To interact with the robot, users can communicate through a device using text and visuals. The robot features a natural language processing component that can understand instructions, ask for clarification when needed, and use this information to execute tasks. When the robot receives unclear instructions, it asks the human for clarification before carrying out tasks, such as moving to a location, manipulating objects, or providing visual descriptions.

2.0 Mission Planning:

As depicted in Figure 1 (2022) [8], the mission planner component receives a stream of actions, models, and constraints derived from the Natural Language Understanding module. This interaction highlights the critical role of the intelligence architecture in translating human input into actionable directives, demonstrating the system's ability to bridge the gap between human intent and robotic action. The PA-BT within the mission planner processes the input and transmits a behavior tree to the mission executor for execution.

The mission planner and executor work in tandem, with the mission executor primarily responsible for ensuring the successful execution of task actions. In contrast, the mission planner serves as the architect, designing the behavior tree that acts as the blueprint for task execution. Once the robot comprehends the task through the language model, it must coordinate and initiate the requisite behaviors for task execution. In the methodology proposed by Thomas Howard and his colleagues in 2022 [8], they employ a variation of the Planning and Action Behavior Tree (PA-BT) algorithm initially introduced by Colledanchise and Ögren in 2018 [4].

3.0 Introducing The PA-BT:

The Planning and Acting using Behavior Trees (PA-BT) approach is originally detailed in Michele Colledanchise and Petter Ögren's paper *Behavior Trees in Robotics and AI: An Introduction* (2018) [4]. It can be found starting on page 94.

Section 3.1: Finite vs. Infinite State Space:

In traditional planning algorithms, the world is typically perceived as static, with all parameters assumed to remain constant (2018) [4]. This approach proves ineffective in real-world scenarios characterized by unpredictability and continuous change. This is where the Planning and Action Behavior Tree (PA-BT) algorithm comes into play, as it excels in operating within an infinite state space.

Section 3.2: The Challenge of Infinite State Space

An infinite state space presents a complex landscape filled with countless variations and conditions, rendering it impractical to enumerate them all. To illustrate this complexity, imagine trying to count every decimal number between 0 and 1—an unfeasible task that mirrors the intricacies of our ever-changing world.

Section 3.3: Robotics and State Space

Consider a robot operating in a finite state space, instructed to open a door. If an obstacle suddenly appears in its path, a robot confined to a finite state space will follow its pre-programmed instructions, even if it's no longer appropriate. This could result in the robot attempting to walk in place and then trying to open a door that is not within its reach. In contrast, a robot equipped with the PA-BT can instantaneously adjust its behavior in response to changing conditions.

For example, it can quickly clear an obstructed path before proceeding. This adaptability is a direct result of the behavior tree structure. As we delve further into how the PA-BT operates within an infinite state space using behavior trees, we will demonstrate how it empowers robots to make real-time adjustments in response to dynamic environmental changes.

4.0 Demystifying the PA-BT:

The PA-BT encompasses two core algorithms, specifically Algorithms 5 and 6, from Michele Colledanchise and Petter Ögren's paper *Behavior Trees in Robotics and AI: An Introduction* (2018) [4]. Both algorithms can be found on page 97. This section describes the impact and functionality of these algorithms.

Section 4.1: Background

The PA-BT was inspired by the Hybrid Backward Forward (HBF) Algorithm, which was developed by Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling in their paper *Backward-forward search for manipulation planning* to address the challenges of large state spaces, including infinite state spaces (2015) [1]. It uses reachability graphs where each edge is an action, and the connected node is the resulting effect of the action. The PA-BT takes the idea of a reachability graph for its ability to work in infinite state spaces and uses behavior trees to add more reactability to its design.

The PA-BT was also influenced by the Hierarchical Planning in the Now (HPN) algorithm, as outlined by Kaelbling and Lozano-Pérez in their paper *Hierarchical Task and Motion Planning in the Now* (2011) [3]. The HPN focuses on the step that is currently happening rather than planning a long list of elaborate actions. Planning out every future action when a slight environmental change can alter the task plan is computationally inefficient and wasteful.

The core concept of Behavior Trees (BT) is to substitute conditions with compact behavior trees that fulfill those conditions (2018) [4]. For instance, in the scenario described in section 3.3, when the robot's path becomes obstructed, and the "path clear" condition is no longer satisfied, the tree seamlessly adjusts to this change in conditions. It expands to incorporate the necessary actions to meet the "path clear" condition, allowing the robot to return to its task.

Section 4.2: The Scenario

To demonstrate the algorithm, we will use a simple scenario. Much like the prior example in section 3.3, the robot's objective is to open a door on the opposite side of the room. It's essential to emphasize that the algorithm allows for operation within an infinite state space so we can adapt to environmental changes. The initial behavior tree, seen in Figure 2.0, consists of a single node with the postcondition, c, that must be met for our goal to be reached. The postcondition in this scenario would be an open door. c = door open.

This idea can also be expanded to a group of goals, where our initial goal and precondition, denoted as ‘c,’ are elements within the set of goals, represented as ‘c ∈ {goals}.’ In this context, our final condition would encompass the entirety of the set of goals, ensuring that the entire set, denoted as ‘{goals},’ is achieved rather than exclusively ‘c.’

Figure 2.0 is our initial behavior tree with our initial precondition or goal.

Following the initial iteration, the robot detects that the door remains closed, prompting the tree's expansion. We begin each time step by revisiting the behavior tree's root., now transformed into a fallback node serving as a substitute for the previously unmet condition.

In this updated configuration, we introduce two new action sequences, as visually depicted in Figure 2.1. The "?" symbol represents our fallback node, while the arrow symbol "→" denotes a sequence encompassing specific actions and their preconditions. To achieve our postcondition, "door open," the robot can opt for either of the two actions, "push" or "pull." This dynamic allows for adaptable decision-making within the behavior tree framework.

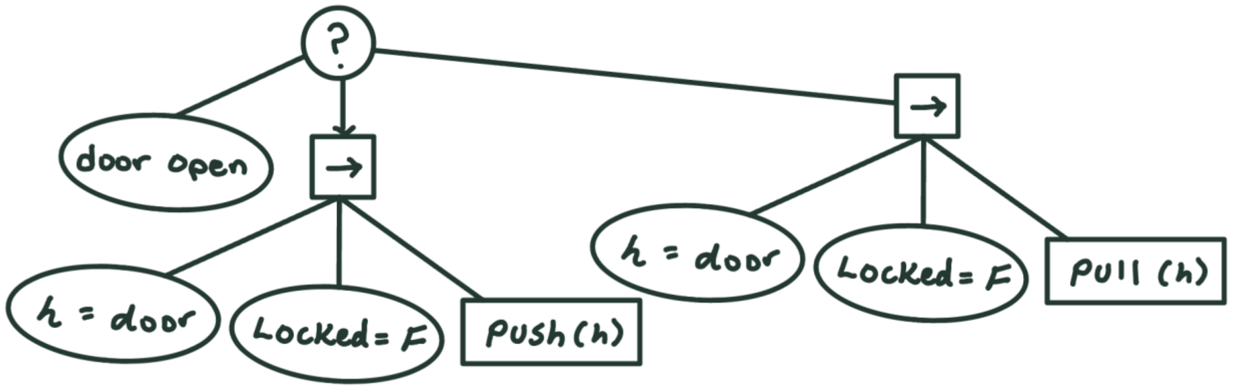


Figure 2.1 The behavior tree after *one* time step.

In Figure 2.1, *"h"* represents an object within hand's reach. "Locked = F" is a condition that verifies the door is not locked, ensuring its status is false. Our primary objective remains to open the door, which can be achieved by following either action sequence. Initially, the robot initiates the "push(*h*)" sequence. If the preconditions are satisfied but the door remains closed, it switches to the second sequence, "pull(*h*)." In this context, we'll assess whether the preconditions for "push(*h*)" have been met. In this specific scenario, the robot checks if the object represented by "h" is the door and whether it is within hand's reach. Given that the robot starts on the opposite side of the room, it identifies that this precondition has not yet been fulfilled. When the condition "h = door" yields a failure, the behavior tree expands once again.

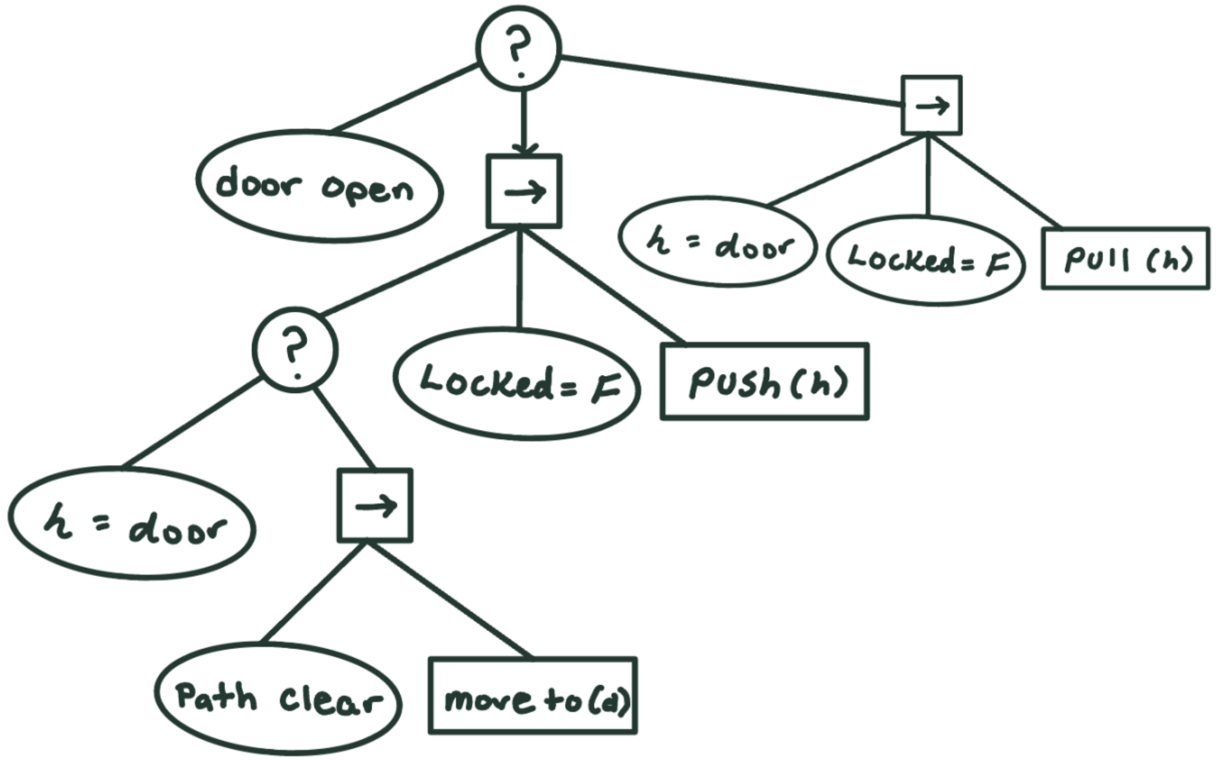


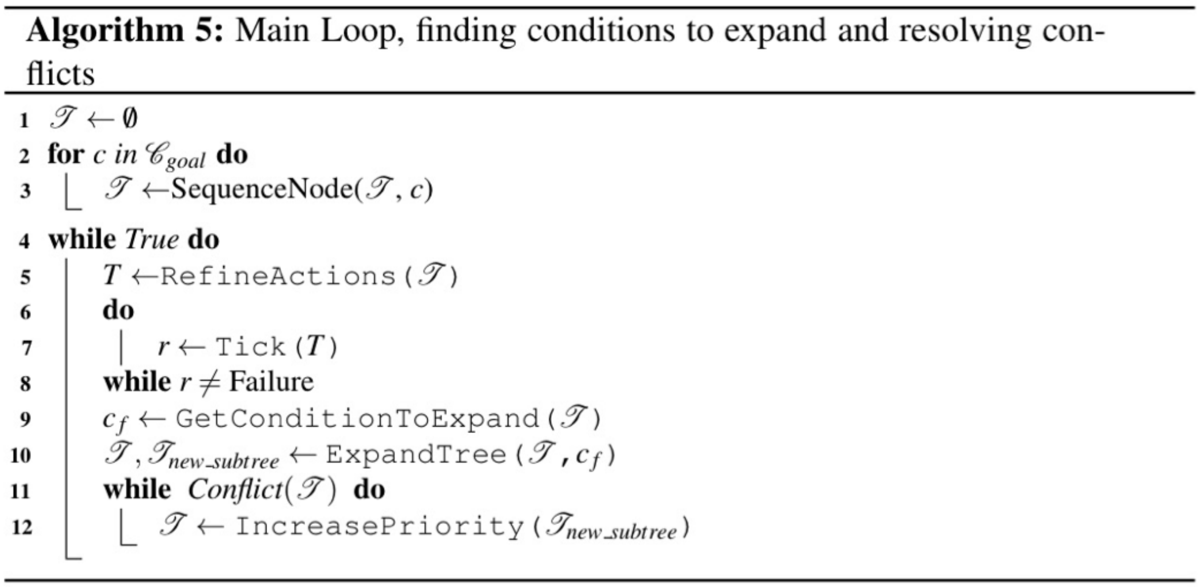
Figure 2.2 The behavior tree after *two* time steps.

Our original precondition, “*h* = door,” now transitions into its own postcondition, leading to the introduction of a new fallback node represented by “?.” Additionally, we incorporate a new action sequence, “move to (d),” where “d” signifies the door. This iterative process of incorporating subtrees and fallback nodes continues until our initial goal or set of goals is successfully achieved.

In this scenario, the process concludes once the door is successfully opened. If, at any point, an obstruction in the path disrupts the “path = clear" precondition, the behavior tree expands to include a new action sequence aimed at re-establishing the “path = clear” condition. This aligns with the algorithm's core structure, guaranteeing precondition verification before executing any action. This means that if our environment changes unpredictably, planning is not redone entirely, and the algorithm structure seamlessly accounts for plan alterations.

Section 4.3: The Algorithms

Algorithm 5 from Michele Colledanchise and Petter Ögren’s paper (2018) [4], as shown in Figure 3.0, denotes an algorithm that looks for conditions that need to be expanded into action sequences.



Algorithm 5: Main Loop, finding conditions to expand and resolving conflicts

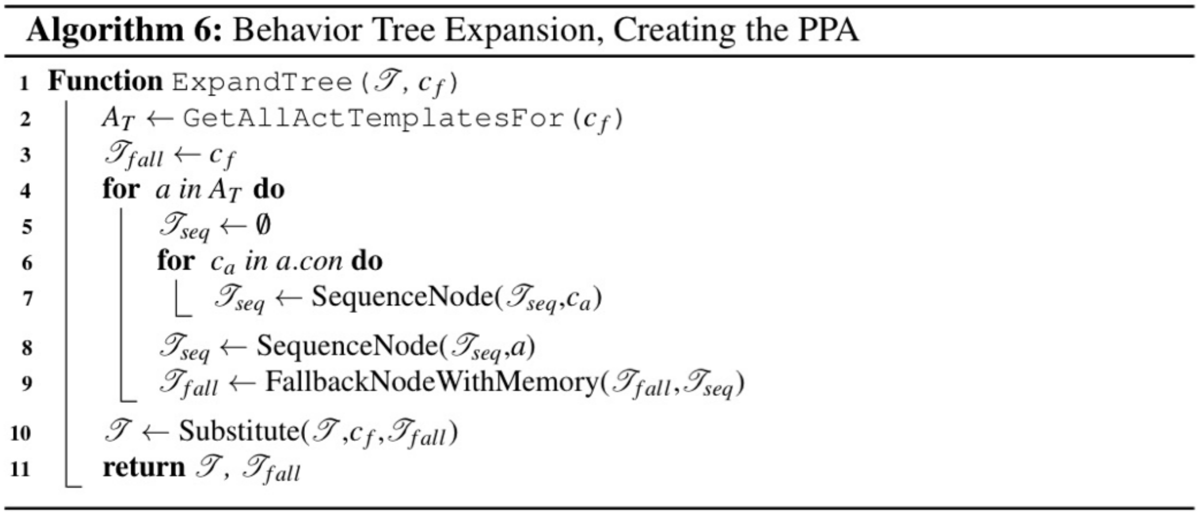
```
1  $\mathcal{T} \leftarrow \emptyset$
2  for $c$ in $\mathcal{C}_{goal}$ do
3      $\mathcal{T} \leftarrow$ SequenceNode($\mathcal{T}$, $c$)
4  while True do
5      $T \leftarrow$ RefineActions($\mathcal{T}$)
6      do
7          $r \leftarrow$ Tick($T$)
8      while $r \neq$ Failure
9      $c_f \leftarrow$ GetConditionToExpand($\mathcal{T}$)
10     $\mathcal{T}, \mathcal{T}_{new\_subtree} \leftarrow$ ExpandTree($\mathcal{T}$, $c_f$)
11     while Conflict($\mathcal{T}$) do
12         $\mathcal{T} \leftarrow$ IncreasePriority($\mathcal{T}_{new\_subtree}$)
```

Figure 3.0 Algorithm 5, see (2018) [4].

1. **Initialization (Line 1):** The algorithm initializes by creating an empty tree, denoted by $\mathcal{T}$. This tree will be used to organize the processes in a hierarchical manner.

2. **Goal Conditions Setup (Lines 2-3):** It then processes each goal condition from a set of goal conditions, c_{goal}. For each condition c, the algorithm adds a sequence node to the tree $\mathcal{T}$. This node represents the sequence of actions and conditions necessary to achieve the goal, "→."”
3. **Main Loop Start (Line 4):** The algorithm enters a loop that will run indefinitely. The subsequent steps are repeated within this loop until the process is manually stopped or a break condition is met.
4. **Action Refinement (Line 5):** The Refine Actions function looks for action templates to add to the tree (2018) [4].
5. **Action Execution (Lines 6-8):** This process moves us through the tree as long as ticking r does not return a failure. That is, it activates nodes until it produces a failure.
6. **Condition Expansion (Lines 9-10):** When a condition requiring further expansion is identified, the algorithm expands the tree $\mathcal{T}$ with a new subtree corresponding to the condition, c_{goal}. This expansion is like adding more branches to the tree for additional options or actions.
7. **Conflict Resolution (Lines 11-12):** While there are conflicts within the tree $\mathcal{T}$, the algorithm works to resolve these by increasing the priority of the added subtree. The algorithm tries to resolve conflicts by rearranging or reprioritizing the nodes within the tree.

An important feature to highlight is line 11 of algorithm 5. This line acts as a checkpoint, identifying potential conflicts where a new action could inadvertently undermine the precondition of another action. At line 12, the algorithm increases the priority of the new subtree sequence. Moving the tree to the left will ensure the action is carried out before the precondition it would undo. This is a crucial step to ensure the system's robustness and reliability. The tree structure in this algorithm serves as a way to organize and handle complex sets of actions and conditions, constantly updating and resolving conflicts to adhere to the defined goal conditions.



Algorithm 6: Behavior Tree Expansion, Creating the PPA

```
1  Function ExpandTree($\mathcal{T}$, $c_f$)
2      $A_T \leftarrow$ GetAllActTemplatesFor($c_f$)
3      $\mathcal{T}_{fall} \leftarrow c_f$
4      for $a$ in $A_T$ do
5          $\mathcal{T}_{seq} \leftarrow \emptyset$
6          for $c_a$ in $a.con$ do
7              $\mathcal{T}_{seq} \leftarrow$ SequenceNode($\mathcal{T}_{seq}$,$c_a$)
8          $\mathcal{T}_{seq} \leftarrow$ SequenceNode($\mathcal{T}_{seq}$,$a$)
9          $\mathcal{T}_{fall} \leftarrow$ FallbackNodeWithMemory($\mathcal{T}_{fall}$,$\mathcal{T}_{seq}$)
10     $\mathcal{T} \leftarrow$ Substitute($\mathcal{T}$,$c_f$,$\mathcal{T}_{fall}$)
11     return $\mathcal{T}$, $\mathcal{T}_{fall}$
```

Figure 3.1 Algorithm 6, see (2018) [4].

Algorithm 6, detailed in Colledanchise and Ögren's work [3], is responsible for seamlessly crafting and integrating action sequences into the Behavior tree. The Postcondition-Precondition-

Action (PPA) being created here is the action sequence. Algorithm 6 can be seen in Figure 3.1 and has been provided to add context to line nine of Algorithm 5, Figure 3.0.

5.0 Other Approaches to Mission Planning:

Section 5.1: Industrial Robots

The PA-BT distinguishes itself from traditional mission planning methodologies, particularly in the industrial domain, where robots are known for precision but are limited in their adaptability and interaction with dynamic environments and humans. In contrast, PA-BT thrives in scenarios requiring flexibility and human interaction, pushing its utility beyond conventional industrial uses.

Francesco Rovida et al. present an innovative group mission planning for industrial robots in their paper *Integrating Mission and Task Planning in an Industrial Robotics Framework* (2017) [2]. The study enables task distribution through a logistic manager that makes decisions based on each robot's skills, contrasting with Howard et al.'s focus on individual robotic mission planning. Despite the decentralized planning, individual robots still decide on specific behaviors to execute assigned tasks, drawing parallels to Howard et al.'s intelligence architecture and the PA-BT's tree-like planning structure.

SkiROS, a framework used in this collective approach, leverages algorithms for creating and refining planning domains essential for executing and allocating tasks efficiently in collaborative robot groups (2017) [2]. These algorithms identify skills, predicates, and goals, as well as enhance action capabilities and spatial object relations, underscoring the significance of effective task management in collective robotics.

In summary, PA-BT's adaptability and responsiveness to human interaction and environmental changes make it suitable for various applications. Traditional robots remain adept at fixed tasks in stable settings. At the same time, group-oriented planning approaches like SkiROS offer a different paradigm that resonates with Howard et al.'s architecture, focusing on collaborative planning for robotic groups. Each strategy provides distinct advantages, addressing various robotic application needs.

Section 5.2 Empathetic Robots

Another approach to mission planning is outlined in Yang Tao et al.'s work, which introduces a sophisticated multi-layered interaction architecture for an elderly companion robot designed to provide a multifaceted engagement experience through conversational, emotional, and musical means (2008) [9]. The architecture comprises three integral layers: a proactive process layer for rapid response and real-time interaction, an interactive controlling layer that underpins potential learning and adaptation, and a behavior-generating and outputting layer that operationalizes the robot's actions. Demonstrated through practical experiments in a nursing center, the architecture showcases its efficacy in enriching the lives of aged individuals by fostering interaction, offering emotional support, and promoting mental engagement, thereby addressing the crucial aspect of social isolation in the elderly population.

This approach to mission planning plays a very different role from the logical process of other intelligence architectures. Introducing a multi-layered interaction architecture offers a compassionate approach, emphasizing emotional well-being and companionship. Compared to the PA-BT algorithm, this approach provides a more empathetic and supportive interaction, essential in environments like nursing centers, where addressing social isolation and promoting emotional and mental engagement are paramount. Thus, the emotional-centric approach marks a significant advancement, showcasing the potential of robotics in providing comprehensive and empathetic care in any caregiving environment.

6.0 Personal Reflection

Embarking on this research journey, my understanding of robots was somewhat rudimentary. I perceived them within the narrow confines of their mechanical functions, often in industrial settings. This was the first I had heard of an Intelligence Architecture, and I did not think they were being developed. Of course, Every research group has its own term for its robot's software architecture; they aren't all intelligence architectures, but I found the term Intelligence Architecture to be specifically interesting. In previous semesters, I studied motion planning for intelligent robots. Still, it never included a comprehensive model that allowed for communication with humans or gave the robot a rich semantic understanding of its world.

Since introducing myself to these architectures, I have anticipated the design of emotionally empathetic technology. I want to see the integration of emotional empathy into personalized AI, such as the virtual personal assistant Jarvis from the Marvel Cinematic Universe. Perhaps I shouldn't wait and instead should start development. If it can be researched, why not explore it myself?

I also learned that behavior trees were initially created for video game characters. I found it fascinating how art can improve science and inspire innovation. More can be learned from the gaming sector on developing effective and dynamic systems. I could learn from both strengths and flaws in their work to adopt or avoid respectively.
This academic endeavor has enriched my knowledge and refined my ability to engage in meaningful discourse about AI and Robotics. Once daunting, the complexities of the topics now present a challenge that I approach with confidence. I appreciate the intricate layers of design, function, and integration of robots in various sectors, recognizing the symbiotic relationship between AI advancements and robotic applications. I also now have multiple sources in my toolbox to reflect on in future studies.

Initially, I thought my main interest was in robots, but now I realize I'm more excited about creating intelligent agents, whether they're robots or not. My research into this subject has also opened my eyes to other creative uses of robotics, like machines that are inspired by animals. I would love to create robotic cats one day. I'm fascinated by how cats move so smoothly and always land on their feet. Building those features into a robot and developing the software to make it work would be amazing. The idea of creating such intelligent agents is appealing to me.

7.0 Conclusion

In conclusion, the field of mission planning in robotic systems has witnessed significant advancements, with the Planning and Action Behavior Tree (PA-BT) algorithm serving as a remarkable milestone in the realm of robotics intelligence. Howard et al.'s work (2022) [8], based on the research by Colledanchise and Ögren (2018) [4], exemplifies the innovative potential of the PA-BT approach. This dynamic solution offers a strategic response to the challenges posed by real-world, ever-changing environments, marking a departure from traditional planning algorithms designed for finite state spaces. The PA-BT's forte lies in its exceptional performance within infinite state spaces, enabling robots to adapt instantaneously to unforeseen conditions.

This adaptability is rooted in the unique structure of behavior trees, where sequences of actions are continuously evaluated and modified to achieve predefined objectives. Drawing inspiration from the Hybrid Backward Forward (HBF) and Hierarchical Planning in the Now (HPN) algorithms, the PA-BT emerges as a powerful tool for autonomous mission planning in the field of robotics. It not only enhances reactability and efficiency but also showcases the ability to tackle complex, unpredictable scenarios. As a result, the PA-BT algorithm represents a significant leap toward the development of autonomous and intelligent robotic systems, fully equipped to navigate and respond effectively to the dynamic challenges presented by their environments.

In the broader context of mission planning, our comparative analysis underscores the remarkable adaptability of the PA-BT, positioning it as a compelling choice for situations that necessitate effective human-robot collaboration and adaptability within dynamic, unstructured environments. While traditional industrial robots continue to excel in precision manufacturing tasks, the PA-BT's exceptional ability to operate within infinite state spaces and as an individual unit makes it stand out.

References

[1] Caelan Reed Garrett, Leslie Pack Kaelbling and Tomás Lozano-Pérez. (2015) 'Backward-forward search for manipulation planning.' In Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference.

[2] Francesco Rovida *et al*. (2017) 'Integrating Mission and Task Planning in an Industrial Robotics Framework.' Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling.

[3] Leslie Pack Kaelbling and Tomás Lozano-Pérez. (2011) 'Hierarchical task and motion planning in the now.' In Robotics and Automation (ICRA), 2011 IEEE International Conference.

[4] Michele Colledanchise and Petter Ögren. (2018) 'Behavior trees in robotics and AI: An introduction,' CRC Press.

[5] Mike Daily *et al*. (2017). 'Self-Driving Cars,' Institute of Electrical and Electronics Engineers (IEEE), Volume 50.

[6] Robin R. Murphy (2019). 'Introduction to AI Robotics,' Second Edition. MIT Press.

[7] Stuart Russell (2021). 'The History and Future of AI,' Oxford Review of Economic Policy, Volume 37.

[8] Thomas M. Howard *et al.* (2022) ‘An intelligence architecture for grounded language communication with Field Robots,’ Field Robotics.

[9] Yong Tao, *et al.* (2008). ‘A Multi-layered Interaction Architecture for Elderly Companion Robot.” International Conference on Intelligent Robotics and Applications (ICIRA).