



UNIVERSITY OF GENOVA

DEPARTMENT OF INFORMATICS, BIOENGINEERING, ROBOTICS, AND
SYSTEMS ENGINEERING

Developing a Social Robot Capable of Changing Unhealthy Habits using Theory of Mind and Nudging Capabilities

by

Giulia Berettieri

October 2024

Supervisors:

Antonio Sgorbissa, *RICE lab, University of Genoa*

Carmine Recchiuto, *RICE lab, University of Genoa*

Lucrezia Grassi, *RICE lab, University of Genoa*

Supervisor

Co-Supervisor

Co-Supervisor

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

Vorrei dedicare questa tesi ai miei nonni, Irene e Danilo.

Loro mi sono stati vicini fin dai primi passi, portandomi per mano nel mondo della scienza.

Tra le gite nei boschi insieme al nonno, dove mi spiegava le affascinanti curiosità della natura, e le lunghe conversazioni della nonna che ha sempre parlato di grandi matematici e filosofi come suoi amici e mentori, hanno creato il connubio perfetto.

Loro mi hanno ispirata a migliorarmi sempre conoscendo sempre più, alimentando la mia curiosità e mostrandomi anche nei momenti più difficili a non arrendersi mai.

Acknowledgements

I would like to acknowledge...

My supervisor, Antonio Sgorbissa, who has guided me from the moment this thesis was proposed, and my co-supervisor, Lucrezia Grassi, who has supported me in developing the technical aspects of the project.

PhD candidate Allegra Bixio, who made an incredible contribution to the psychological aspects of this work and who is proof that the union of different fields can lead only to great results. In fact, we were an amazing team and had the opportunity to present our work at two internationally renowned conferences, such as ROMAN and ICRA@40 (*Trust the process*).

My family, especially my mother Erica and my father Davide, who have supported and encouraged me, sharing my fascination and curiosity for the world of robotics. They have shared my obsessions and have made valiant attempts to keep the peace to understand where in the world I had wandered off to.

Natan, who has supported me in every decision and never let me give up, has stood by my side even in the scariest choices, providing me with endless courage and support. Thank you for sharing so many laughs and for exploring the world together, even if sometimes the best adventures are just a simple chat on the couch.

My inseparable study companions, Tabi and Franciccina, who I happened to meet in the first few days and without whom I would not have made it here. Together, we created a new path to follow, and I could not be happier to continue this journey with them.

The lifelong friends, in no particular order, Irene, Camilla, Deborah, Arianna, Mattia, and Nicole, who listened to my long complaints, and were always ready to offer solutions. Despite the passing of years and the diverging paths we've taken, they remind me that the bond we have shared keeps us connected.

The RICE Lab, GRAAL, and MacLab, and all those who are part of these groups, for welcoming me with open arms and for their patience. I am excited to be part of this family.

Abstract

Integrating social robots into everyday life holds great potential for influencing human behavior, particularly in therapeutic and educational settings. This thesis explores the development of a social robot capable of recognizing and managing challenging behaviors during child-robot interaction, specifically applying principles of Theory of Mind (ToM) and Applied Behavior Analysis (ABA).

The core objective of this work is to create a robotic system that can autonomously adapt to real-time human interactions, enhancing engagement and providing personalized interventions.

The Planning Domain Definition Language (PDDL) framework is at the heart of the system, which facilitates the robot's ability to understand, plan, and execute actions based on human behavior. The architecture integrates advanced cognitive models, behavior management strategies, and natural language processing through OpenAI services, allowing the robot to recognize mental states, identify behavioral patterns, and apply corresponding intervention strategies.

Results from the implementation demonstrate that the robot can successfully adapt to dynamic environments, recognize user needs, and influence behavior in real time. This framework lays the groundwork for future advancements in social robotics. Future work will focus on optimizing the system's components to enhance efficiency and user experience.

Table of contents

List of figures	vii
List of tables	viii
Nomenclature	ix
1 Introduction	1
1.1 Context	1
1.2 Objectives	2
1.3 Results	3
1.4 Document structure	3
2 State of the Art	5
2.1 Motivation	5
2.2 Theory of Mind in Robotics	7
2.3 Behavior Change Techniques and Nudging	11
2.4 Educational Robotics	15
2.5 Applied Behavior Analysis in Robotics	16
2.6 Planning in HRI	19
3 Methodology	21
3.1 General Architecture Overview	21
3.2 Implementation of PDDL	23
3.2.1 Planner Engine	23
3.2.2 Theory of Mind	24
3.2.3 Behavior Functions and Strategies	25
3.3 Action Implementation	30
3.3.1 Actions that do not require execution	30

3.3.2	Actions that can be scripted	30
3.3.3	Interactive actions	32
3.3.4	Prompt Design	32
3.4	Server-Client Model	34
3.4.1	Uploads	35
3.4.2	Client	36
3.4.3	Server	38
3.5	Implementation Process	40
4	Framework Evaluation	42
4.1	Experimental Design	42
4.2	Measurements and Data Collection	44
4.3	Data Representation	52
5	Conclusion	55
5.1	Summary of Findings	55
5.2	Future Directions	56
	References	57
	Appendix A Detailed Description of PDDL Domain Elements	62
A.1	Predicates	62
A.1.1	Agent State Predicates	62
A.1.2	Robot State Predicates	63
A.1.3	Task State Predicates	63
A.1.4	Tangible State Predicates	63
A.1.5	Theory of Mind Predicates	63
A.2	Actions	65
A.2.1	Interact	65
A.2.2	PresentTask	65
A.2.3	Wait	65
A.2.4	WarningCleanUp	65
A.2.5	PutAway	66
A.2.6	ConfirmAllGood	66
A.2.7	GoodJob	66
A.2.8	GoodJobStrategy	66

A.2.9 DiscoverTOM1	67
A.2.10 DiscoverTOM2	67
A.2.11 ConfirmHardTask	67
A.2.12 StrategyHardTask	68
A.2.13 ConfirmBoringTask	68
A.2.14 StrategyBoringTask	68
A.2.15 ConfirmWant2Play1	68
A.2.16 ConfirmWant2Play2	69
A.2.17 StrategyWant2Play	69
Appendix B Experimental Protocol	70
B.1 Participants	70
B.2 Hypotheses	70
B.3 Experimental Procedure	71
B.4 Measurements	72

List of figures

2.1	Integration SDT in basic motivational model (1).	6
2.2	Baron Cohen's ToM model (2).	9
2.3	False belief task illustrated.	10
2.4	Everyday nudging examples.	12
2.5	Social cues.	14
2.6	Functional assessment methods.	17
2.7	Four functions of behavior.	18
2.8	Hanoi tower (3).	19
3.1	Architecture overview.	21
3.2	Fast Downward phases (4).	23
3.3	Planner overview.	29
3.4	RestAPI model.	34
3.5	Pepper robot.	36
4.1	Experimental setup.	42
4.2	Memory setup.	43
4.3	Cloud response time vs. number of characters in the user's sentence.	54

List of tables

2.1	Children’s motivational responses (5).	7
2.2	Examples of different roles of robots in education (6).	15
3.1	PDDL actions related to behavior management.	26
3.2	PDDL modeled behavioral functions and corresponding strategies.	27
3.3	PDDL actions that do not require execution.	30
3.4	PDDL strategy actions that can be scripted.	30
3.5	PDDL presenting action that can be scripted.	31
3.6	PDDL praising actions that can be scripted.	31
3.7	PDDL warning action that can be scripted.	31
3.8	PDDL interactive actions.	32
4.1	Plan retrieval time (s).	52

Nomenclature

Acronyms / Abbreviations

ABA Applied Behavior Analysis

API Application Programming Interfaces

BCT Behavior Change Technique

CRI Children-Robot Interaction

EDD Eye Direction Detector

ES Emphatizing System

FBA Functional Behavior Assessment

FD Fast Downward

HRI Human-Robot Interaction

HTTP HyperText Transfer Protocol

ID Intentionally Detector

PDDL Planning Domain Definition Language

REST Representational State Transfer

SAM Shared Attention Mechanism

SAR Social Assistive Robot

STD Self-Determination Theory

ToM Theory of Mind

ToMM Theory of Mind Mechanism

URI Uniform Resource Identifier

Chapter 1

Introduction

Behavioral change is inherently complex, even when individuals recognize the necessity of change for their health. The potential for social robots to address these challenges is substantial. Through this research project, I aim to explore the intersection of robotics, psychology, and social behavior. Specifically, I intend to focus my research on understanding how robots can enhance human interactions.

1.1 Context

Social robots, designed to interact with humans in a social context, have the potential to bridge the gap between technology and human needs. A key component in this effort is the role of Social Assistive Robots (SARs). SARs are not only designed to perform tasks but also to assist individuals in the social and emotional aspects of learning and development. They can support diverse populations, including children with special needs, the elderly, and those undergoing rehabilitation.

By integrating Theory of Mind (ToM) capabilities, intended as the capacity to understand other people by ascribing mental states to them, SARs can better comprehend and respond to users' emotional states. This enables them to provide tailored encouragement, feedback, and guidance, significantly enhancing the learning experience. SARs, therefore, have the potential to transform education and care environments by offering personalized, empathetic, and context-aware support.

Different methods and techniques exist to acknowledge mental state, and despite its controversial role as a support for people with neurodevelopmental disorders (7), an ABA-inspired robot capable of reasoning about the purpose of others' behaviors can be useful support in educational settings. Applied Behavior Analysis (ABA) is an evidence-based

approach that explains how behavior works and is affected by the environment, and it is presently used in schools due to its effective behavior management techniques (8). Applying these strategies to social robotics could improve the quality of child-robot interaction (CRI) in learning environments.

Please notice that, understanding how social robots can facilitate behavior change shares some similarities with serious games which utilize engaging and interactive formats to encourage learning and personal development (9). These games can serve as a medium through which social robots promote positive behaviors by creating immersive experiences that encourage participants to reflect on their actions. Although these concepts have inspired parts of this work, they are not the primary focus of this project.

The exploration of the complex dynamics that emerge in CRI necessitates a comprehensive understanding of behavioral modeling, which we choose to model through the lens of Planning Domain Definition Language (PDDL). PDDL offers a structured way to represent actions and their effects, facilitating the development of robots capable of planning and executing behaviors that are responsive to human needs and contexts.

1.2 Objectives

The analysis of the state-of-the-art (Chapter 2) emphasizes the necessity of designing sociable robots capable of integrating into daily life and sustaining user engagement over time. This project aims to fill this gap by developing an architecture that adapts to users' situations and needs. By bridging psychological theory with practical application, this project showcases the potential of social robots to effectively influence and enhance human behavior. The integration of personalized approaches ensures that these robots provide meaningful and beneficial interactions in therapeutic, educational, and everyday contexts.

Moreover, a deeper analysis of the state-of-the-art in behavior management through HRI reveals that robots still lack autonomy in recognizing the goals of other agents and adapting interactions accordingly. Studies inspired by ABA strategies often use methods where the robot's behavior is controlled by an experimenter. Hence, an innovative architectural system based on the PDDL framework is proposed. This architecture enables real-time adaptation and allows the robot to manage interactions autonomously.

The overall contributions of this work are the following:

- Developing a theoretical model that takes into account the constraints in CRI, serving as the robotic counterpart to the ABA model.
- Developing an architecture based on the robotic ABA model previously created. This architecture enables real-time adaptation and allows the robot to form a ToM about the child, grounded in observations and conversations, and plan appropriate actions using the PDDL.
- Evaluating the system's performance in terms of planning and response time, laying the foundation for its future use in CRI.

1.3 Results

It has been demonstrated that the proposed framework works as intended, achieving promising results regarding real-time adaptation and interaction. Among these results, the response time and the planning time, not only validate the system's design but also pave the way for future improvements. Future work will focus on optimizing system components to enhance efficiency and user experience. With these improvements, the potential for sociable robots to deliver meaningful, personalized interactions in various educational and therapeutic contexts becomes even more promising.

1.4 Document structure

The thesis is organized into five main chapters. The first chapter introduces the research problem and outlines the motivations behind this work, focusing on the role of social robots in behavior change. It also defines the objectives and briefly presents the key findings of the study.

The second chapter reviews the state of the art, offering a comprehensive literature survey on essential concepts such as Theory of Mind (ToM), Behavior Change Techniques (BCTs), and Applied Behavior Analysis (ABA), particularly in their applications to robotics. This chapter identifies existing gaps in the field and establishes the framework for the research.

Chapter three describes the methodology employed in this work. It provides a detailed explanation of the system's architecture, including the Planning Domain Definition Language (PDDL) framework, behavior modeling strategies, and the integration of external services

such as OpenAI for real-time human-robot interactions. The server-client model and the various algorithms used to ensure the system's adaptability are also discussed in detail.

In the fourth chapter, the system is evaluated through experimental trials, focusing on its ability to recognize and adapt to user behavior. This chapter presents the results of these experiments, supported by data analysis, and discusses how the system performs across different scenarios.

Finally, the fifth chapter concludes the thesis by summarizing the contributions of this work. It reflects on the outcomes of the research and suggests future directions to improve the system's effectiveness in behavior modification and educational applications.

Chapter 2

State of the Art

This chapter analyzes the current state-of-the-art, identifying existing gaps and outlining how this project aims to address them.

Firstly, Section 2.1 provides a general background on motivation. Two foundational psychological concepts serving as the foundation for this research — Theory of Mind and Nudging Capabilities — are then discussed in Sections 2.2 and 2.3, respectively. These techniques have broad applications across various domains, including elderly care (10), healthcare (11), and more (6).

The primary focus of this thesis is Children-Robot Interaction within the educational field, as detailed in Section 2.4. After considering various theoretical approaches and engaging in extensive discussions with experts, the Applied Behavior Analysis methodology was selected to address behaviors defined as *challenging*. The rationale behind this choice, along with its advantages and disadvantages, is critically examined in Section 2.5.

To put these strategies into practice, the Planning Domain Definition Language was chosen as the framework for implementation. Although this planning language has been previously utilized in Human-Robot Interaction, as reviewed in Section 2.6, its application within this context presents novel challenges and opportunities that are explored in this work.

2.1 Motivation

Thanks to the work of (1), who compiled the most commonly cited theories of motivation in education, it has been possible to gain an overview and a deeper understanding of motivation itself. Motivation is indeed one of the most studied psychological constructs in education, derived from the Latin word *movere*, meaning “to move”, as it provides the energy behind people’s actions. Motivation theories in education have evolved significantly over

time, leading to the establishment of six major theories. This field is dynamic, expanding beyond foundational theories to include situational, social, and cultural factors, insights from neuroscience, the interplay between motivation and emotion, person-centered motivational profiles, and theory-based interventions. To maintain coherence in this expanding research field, a shared understanding of foundational models and theories is crucial.

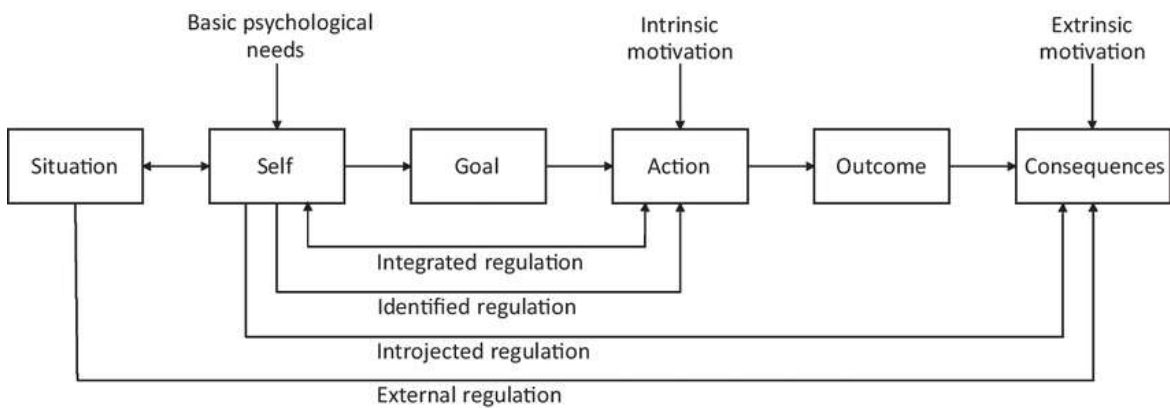


Figure 2.1 Integration SDT in basic motivational model (1).

Among the theories gathered, the Self-Determination Theory assumes in this research a particularly relevant aspect. The representative model shown in Figure 2.1 starts with a Situation that influences the Self, which addresses basic psychological needs. The Self then determines a Goal, leading to Action driven by either intrinsic or extrinsic motivation. The Action results in an Outcome, which has Consequences influenced by extrinsic motivation.

Based on this theory, (12) differentiates between intrinsic and extrinsic motivation to distinguish between behaviors that are self-initiated and those that are driven by external pressures.

Intrinsic motivation refers to engaging in an activity for the inherent pleasure or the satisfaction it provides, rather than for any external rewards or outcomes. When someone is intrinsically motivated, they are motivated by the enjoyment or challenge of the activity itself, not by external incentives, pressures, or rewards. In contrast, extrinsic motivation involves engaging in an activity to achieve a specific, external outcome.

Additionally, (5) conducted a developmental study on children's reasons for liking or disliking certain behaviors, introducing the concept of internalized motivation. This type of motivation refers to behaviors that children may not necessarily enjoy but feel compelled to have because they have been taught and are expected to behave so.

Table 2.1 provides insight into the possible motivational responses based on the interviews they conducted.

Intrinsic	Extrinsic	Internalized
Pursuit of challenge	Anticipation of reward or approval	Self-determined goal achievement
Pleasure or interest in the activity itself	Avoidance of disapproval	Avoidance of self-determined consequences
	Following an explicit rule	Doing something nice for someone else

Table 2.1 Children's motivational responses (5).

Furthermore, (13) conducted an interesting qualitative study on different experiences of motivational interviewing delivered by the NAO robot. Their study aimed to develop an interview protocol within the constraints of current artificial intelligence. The results indicated that social robots can effectively achieve a fundamental objective of motivational interviewing, as they are perceived as nonjudgmental. It is important to highlight how the perception of the interviewer, in this case, the robot NAO, may be influenced by various factors, one of which is synthetic speech generation. According to (14), while human voices are generally perceived more favorably than computer-synthesized voices, there is no evidence that this difference affects the persuasiveness of the message or the perception of the message itself.

2.2 Theory of Mind in Robotics

The ability to reason about other people's perceptions, beliefs, and goals is referred to as Theory of Mind (ToM).

Although these cognitive abilities are fundamental to normal social interaction, they have been extensively studied, not only as a keystone for a functional Human-Robot Interaction (HRI) but also as a critical element of human social cognition. As noted by (2) in his research, for robots to interact naturally with people, they must not only understand the properties of objects and agents but also the complex attributes of animate agents in the world, such as their intentions, beliefs, and desires.

ToM is not the first cognitive milestone in human development though. Rather, it is built upon foundational skills and modules that emerge earlier in life. Theories proposed by Leslie and Baron-Cohen provide a framework for understanding these precursors to ToM.

Leslie's model suggests that infants possess domain-specific cognitive modules that help them understand different types of causality (15):

- *Mechanical*: understanding how physical objects interact;
- *Actional*: interpreting goal-directed actions;
- *Attitudinal*: recognizing beliefs and desires.

This structured approach lays the groundwork for more sophisticated social cognition, where one can infer and predict others' mental states. Leslie proposed that the ability to understand pretense in infancy demonstrates the presence of a specialized form of representation for mental states. This capacity for *meta-representation* enables children to distinguish between reality and imagined scenarios, a skill essential for the development of more complex mental state attributions later in life. By recognizing that others can hold false beliefs or engage in pretend play, children begin to develop a nuanced understanding of the mind, which forms the basis for more sophisticated social interactions and empathy.

Similarly (2), Baron-Cohen's model proposes that infants develop a set of modules that gradually enable them to interpret self-propelled motion, eye direction, shared attention, and ultimately, mental states such as beliefs and intentions. These skills are crucial for developing advanced social cognition, allowing children to navigate complex social environments by attributing mental states to others and understanding that these states can differ from their own.

Therefore, Baron-Cohen's model, as illustrated in Figure 2.2, proposes a developmental pathway for the emergence of ToM in children, progressing through a series of cognitive modules. The Intentionality Detector (ID) and Eye Direction Detector (EDD) are developmentally prior mechanisms, emerging in humans between 0 and 9 months of age. The ID interprets an agent's self-propelled movement as indicative of desires or goal-directed actions, while the EDD is responsive to eye-like stimuli, representing what an agent sees. These early mechanisms provide the foundation for the Shared Attention Mechanism (SAM), which develops between 9 and 14 months of age. The SAM facilitates more sophisticated triadic representations that are crucial for shared social experiences, commonly referred to as the phenomenon of joint attention.

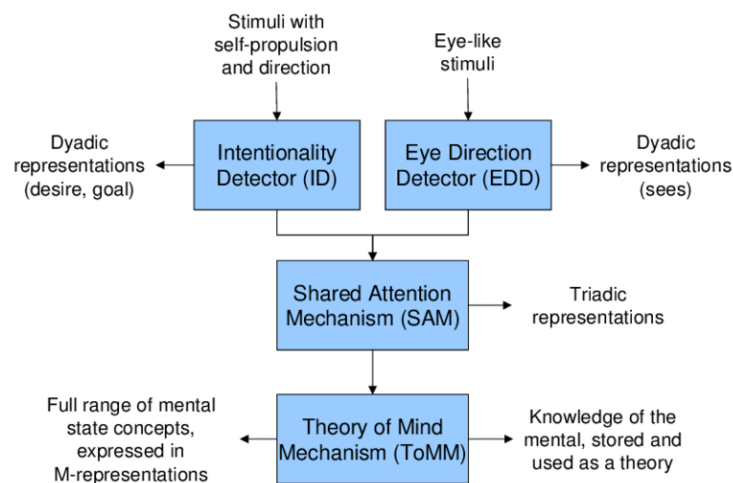


Figure 2.2 Baron Cohen's ToM model (2).

For example:

- ID: "Mother wants the cup."
- EDD: "Mother sees the cup."
- SAM: "Mother sees that I see the cup."

Finally, the Theory of Mind Mechanism (ToMM) enables the representation of epistemic states and complex beliefs, culminating in the full development of ToM. This progression from basic perceptual processing to advanced cognitive abilities aligns with Baron-Cohen's timeline of developmental milestones, providing a structured framework for understanding how children, and potentially robots, could develop the ability to attribute mental states to others.

By implementing such a structured developmental model in humanoid robots, researchers can equip machines with increasingly sophisticated social cognition, enabling them to interact more naturally with humans by recognizing and responding to their beliefs, desires, and intentions. This modular approach also offers insights into specific impairments in social cognition, such as those observed in autism spectrum disorder, where certain modules (like SAM or ToMM) may be selectively impaired. It provides a nuanced framework for understanding the variability in social cognitive abilities across different populations.

In line with these insights, various experiments have been designed to assess human capabilities related to the ToM. One of the most traditional and widely used tests is the *False Belief task*, visually represented in Figure 2.3 (16) (15). This test measures an individual's ability to understand that others can hold beliefs about the world that are different from reality and from the individual's own knowledge. This task is fundamental in evaluating whether a person can recognize that another person may have a belief that is false or different from their own, which is a crucial aspect of the ToM.

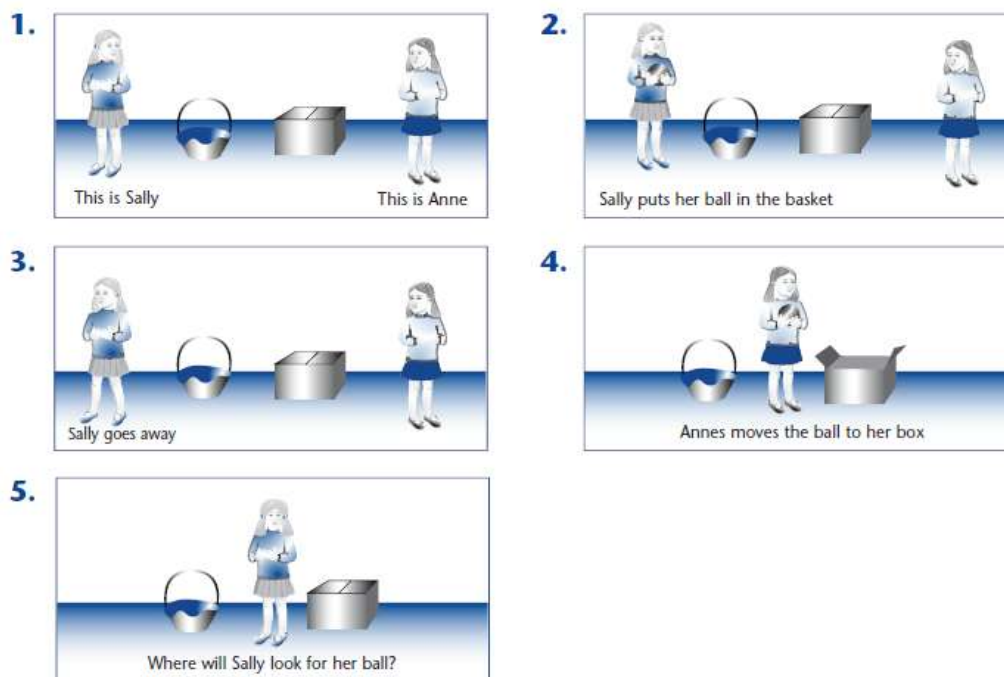


Figure 2.3 False belief task illustrated.¹

The figure is self-explanatory. After presenting the vignette to a child, the interviewer asks, "Where will Sally look for her marble?;" if the child's ToM is fully developed, they will answer "in the basket," understanding that Sally will look where she last saw the marble. This response shows that the child recognizes that Sally has a false belief about the marble's location, as we, the external viewers, know that the marble is actually in the box, but Sally does not.

This task also plays a central role in an innovative robotic framework developed by (16). In this framework, the robot is capable of generating ToM, allowing it to plan sequences

¹<https://autismbeautycombines.wordpress.com/early-signs-of-autism/>

of actions to achieve its own goals based on these representations. This concept will be reviewed in major detail in Section 3.2.

The robotic framework described earlier is just one of the many adaptations and applications of ToM in HRI.

Other notable examples include the work by (17) and (18). In the first one, the researchers focused on developing robots capable of understanding and predicting human actions by modeling ToM capabilities. Their approach involves using advanced machine learning techniques to enable robots to infer human intentions and adjust their behavior accordingly. This is particularly useful in collaborative settings, where robots and humans need to work together seamlessly and intuitively. By anticipating human actions, the robots can make proactive decisions that enhance cooperation and efficiency.

Similarly, (18) explores the application of ToM in robots for social environments, such as healthcare and education. The study investigates the effects of a robot mimicking ToM on human trust and decision-making during collaborative tasks. Their study involves an online experiment where participants interact with a robot (Pepper) in a maze navigation task. The robot's persona varies across conditions—neutral, technical, and ToM—allowing the researchers to assess how mimicking ToM influences human trust and behavior. The findings reveal that robots exhibiting ToM can encourage more cautious and thoughtful interactions, leading to a more appropriate level of trust and better collaboration outcomes.

Even though these examples illustrate the broad applicability of ToM in HRI, showing how robots can be endowed with social cognitive skills to enhance their interactions with humans across various domains, they also highlight the potential for further expansion in this area.

2.3 Behavior Change Techniques and Nudging

Behaviors, including smoking, poor diet, physical inactivity, social interactions, and cognitive engagement play a crucial role in determining health outcomes and our lives. Despite the challenges associated with behavior change, it is possible to achieve significant improvements. This is achievable through Behavior Change Technique (BCT), which is a systematic, theory-based strategy designed to influence specific behaviors by targeting the psychological and environmental factors that drive them.

BCTs are the smallest active components of an intervention, defined as observable, replicable, and irreducible elements that can change behavior by modifying antecedents, a trigger, and consequences (19). BCT taxonomies provide a structured methodology for

characterizing intervention content. For instance, the classification made by (20), which identifies 22 BCTs, has been widely used to report synthesize evidence, and design new interventions. These taxonomies have facilitated research in various domains, such as smoking cessation, physical activity, and healthy eating, by clearly defining intervention components and boosting the effectiveness and consistency of behavior change strategies. This could then be used as an important tool for robotics framework in the future. In recent years, BCTs have been increasingly integrated into technological interventions, including mobile health apps, wearable devices, and robotic systems (21) (22).

A subset of BCTs that subtly influences users' choices and actions without restricting their freedom is nudging. This approach is particularly relevant in educational and social contexts, where robots can be programmed to encourage behaviors that promote learning, social interaction, and well-being.

Nudging, defined as any attempt to influence people's judgment, choice, or behavior in a predictable way, is considered by (23) to be one of the psychological mechanisms most likely to promote well-being. Nudging is even one of the most used behavior change techniques today, with applications observable in everyday life.



(a) Urinal fly.



(b) Ballot bin.

Figure 2.4 Everyday nudging examples.

Figure 2.4 illustrates a few common examples of nudging. On the left, we see the men's restroom in Amsterdam's Schiphol Airport, where in the early 1990s the simple application of

a small fly sticker inside urinals significantly reduced the workload of cleaning staff obtaining an 80% spillage reduction. Another example is the common sight of a trash bin or cigarette disposal, a so-called ballot bin, that encourages proper use by offering the opportunity to “vote” by throwing waste into specific compartments.

With an understanding of what nudging entails, it becomes important to explore whether a robot can effectively implement nudging techniques. Aimed to answer this question, (24) conducted a unique study that involved building one of the first robotic copies of a real human. Indeed, the persuasiveness of the latest technology has become increasingly important as the focus of a new field of research. However, the research primarily concentrated on the embodiment of an agent capable of persuading others.

The literature contains numerous practical case studies, several of which are particularly relevant and representative of behavioral change in robotics applications. Some notable examples include (25), which focuses on motivating children in vegetable consumption. The researchers designed a robot that interacted with individuals, offering fruits and engaging them in conversations about the benefits of healthy eating. The robot utilized various strategies, including reminders, encouragement, and positive reinforcement, to influence the participants’ choices. The study demonstrates that robotic interventions could effectively promote healthier eating behaviors, particularly when the robot is perceived as engaging and supportive. The hand washing behaviors with a social robot specifically designed to promote proper hand hygiene, which is critical for preventing the spread of diseases is addressed by (11). The robot, placed in the restroom, provided instructions on effective hand-washing techniques and offered feedback based on the user’s actions. The interactive nature of the robot, combined with its ability to provide real-time guidance, made it a powerful tool for encouraging compliance with hand hygiene practices. The results indicate that the robot was successful in improving hand-washing behavior. Finally, (26) examines the use of robots in encouraging recycling behaviors. The robot Pepper, programmed to be both informative and interactive, engaged with users by explaining the importance of recycling and guiding them on how to properly sort their waste. The robot’s embodiment, including gestures and social cues, played a significant role in its ability to influence behavior. The study found that participants were more likely to engage in recycling activities when guided by the robot, highlighting the potential of robotics in promoting environmentally responsible behaviors.

What makes (24) particularly intriguing compared to other practical case studies previously mentioned is the advanced level of embodiment achieved. This embodiment introduced a wide range of social cues into the research, including gestures, gaze, posture, and the

overall physical presence of the robot. These social cues are invaluable, as they play a crucial role in human robot interaction.

A different study highlights the importance of social cues, which are defined as the subtle signals, such as body language and eye contact, that facilitate communication and influence perception and behavior in social contexts as highlighted in Figure 2.5. In particular, (27) emphasizes the importance of both verbal and nonverbal cues through a carefully designed experiment. This study demonstrated how robots that effectively utilize a combination of spoken language and physical gestures can significantly enhance user engagement and communication.

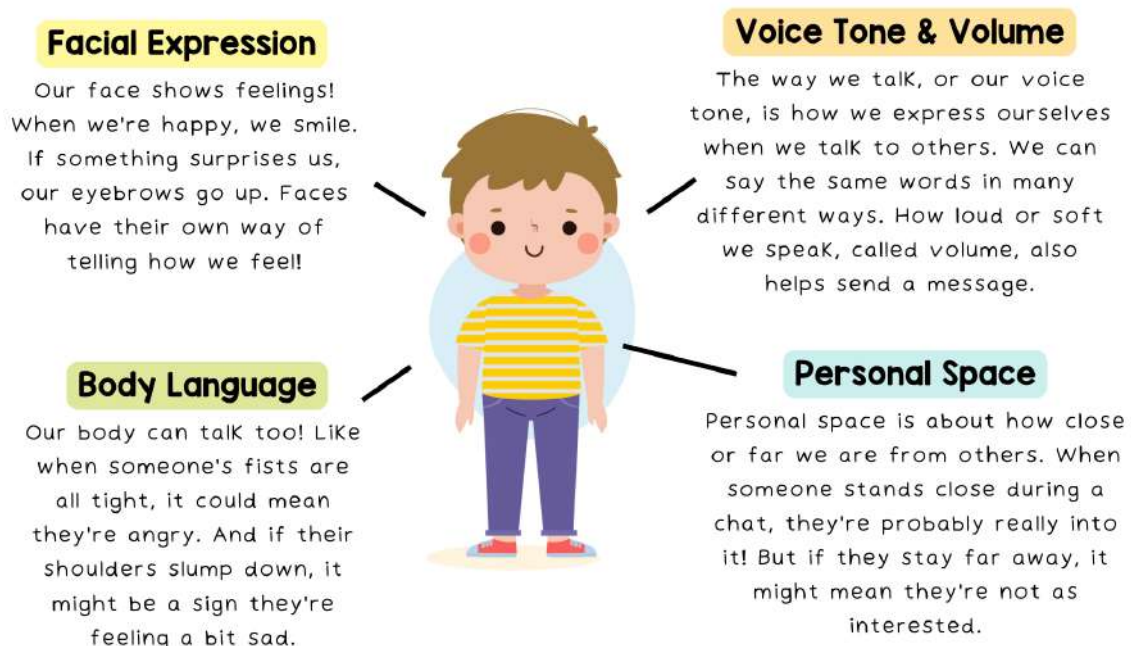


Figure 2.5 Social cues.²

Similarly, (28) explores the relationship between gaze and gesture, highlighting how their synchronization can enhance storytelling. The study found that when robots combined gaze direction with appropriate gestures, the storytelling experience became more immersive and impactful for the audience. This correlation between gaze and gesture not only improved the clarity of the narrative but also increased the overall persuasiveness of the robot, demonstrating the powerful role that nonverbal communication plays in HRIs.

²<https://www.mylemarks.com>

2.4 Educational Robotics

Educational robotics involves using robots as tools to support and enhance the learning process. Over the past few decades, educational robotics has evolved significantly. Initially, robots were seen as mere tools for teaching programming or engineering. However, their role has expanded to include more innovative applications across various educational levels.

As noted by (29), the incorporation of robots into the learning process can lead to innovations at different levels of effectiveness, enhancing both the instructional methods and the learning experience. In particular, the use of robots may increase attention, engagement, and compliance, which are critical components of successful learning (30).

It is important to notice that, the potential of robots to improve educational outcomes extends beyond simply assisting with tasks. According to (31), robots have proven effective in increasing both cognitive and affective outcomes in learners, underscoring their potential as more than just mechanical tools.

Moreover, robots can take on various roles in educational settings, each offering unique benefits. They can serve as tutors, peer learners, or motivational aids, each contributing to different aspects of the learning process as noticeable in Table 2.2. Robots are capable of personalized interaction, allowing them to adapt to each student's individual learning needs.

Tutor	Peer	Tool
The robot adapts the arithmetic exercises based on the performance of the student	The robot and the student collaboratively solve exercises	Sensors and actuators in the robot enable the students to learn

Table 2.2 Examples of different roles of robots in education (6).

Integrating robotics into the curriculum offers numerous benefits, particularly in terms of student engagement and motivation. As noted by (32), robots can provide personalized one-on-one tutoring, producing results comparable to the *2 Sigma effect*, also known as *Bloom's 2 Sigma Problem*, which refers to the significant improvement in educational outcomes that can be achieved through one-on-one tutoring.

Even though robots may not fully replicate the nuances of human instruction, they have demonstrated significant positive effects on student performance. Furthermore, the use of robots in education can motivate interest in subjects that students may initially find difficult or unappealing. For instance, robots can be programmed to adjust their teaching strategies based on a child's performance, enhancing motivation and persistence. This flexibility allows for

more effective and engaging learning experiences, particularly in subjects like mathematics and science that often pose challenges to students (33).

Fundamentally, educational robotics is reshaping traditional learning environments by enabling experiential learning that is tailored to individual students. Robots can, indeed, adapt to the individual needs and learning styles of students, making learning more tailored and effective. In a study by (34), robots were used in an elementary school setting to build long-term relationships with students by playing chess. The study found that these personalized interactions not only improved students' cognitive abilities but also helped foster emotional connections with the robots. Similarly, (35) explored how robots can adapt the difficulty level of educational games to maintain student engagement. By personalizing the learning process in real-time, robots ensure that students remain challenged, but not overwhelmed, which is key to sustained interest and success in learning

2.5 Applied Behavior Analysis in Robotics

The novel robotic architecture for behavior management in educational settings proposed in this thesis, is inspired by Applied Behavior Analysis (ABA), an evidence-based approach that aims at explaining how behavior works and is affected by the environment (36). ABA provides a robust theoretical foundation for understanding and modifying behavior, making it an ideal model for the development of adaptive robotic systems.

In the ABA framework, behavior is considered purposeful (*functional*) and is influenced by stimuli that occur before (*antecedents*) and after the behavior (*consequences*). Certain consequences, referred to as reinforcements, can increase the likelihood of the behavior recurring. Indeed, according to (37), ABA therapy, rooted in behavioral theory principles, emphasizes the connection between behaviors and their environmental consequences.

The ABC Model serves as the cornerstone of ABA. This framework posits that behavior is influenced by three key elements:

- *Antecedent*: the event or stimulus preceding the behavior;
- *Behavior* itself;
- *Consequence*: the outcome that follows the behavior.

By analyzing these components, therapists can identify and modify factors that either maintain or reduce a particular behavior.

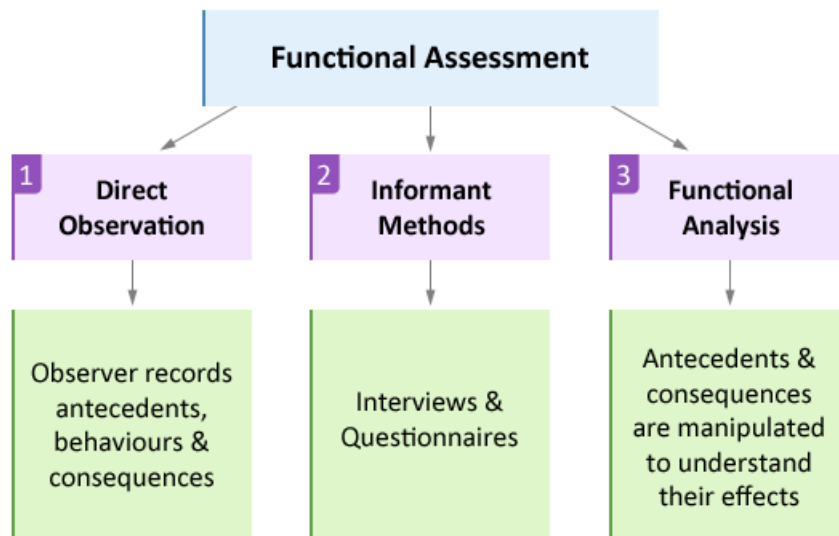


Figure 2.6 Functional assessment methods.³

Moreover, functional assessment is a key component of ABA therapy, used to identify the purpose of challenging behaviors and develop effective interventions. As stated in Figure 3.2, it typically involves three methods: direct observation, informant methods, and functional analysis, where antecedents and consequences are manipulated to assess their impact on behavior. These methods help provide a clear understanding of environmental factors that influence behavior, guiding targeted interventions. Functional Behavior Assessment (FBA) is the process of identifying the functions or purposes of behavior through an analysis of its antecedents and consequences (38), (39). The four predominant functions of behavior are *Gain Attention*, *Escape*, *Gain a Tangible*, and *Sensory Needs* as shown in Figure 2.7.

These functions help us understand and categorize an individual's actions, enabling us to determine why certain behaviors occur. The study focuses on two of these four recognized functions:

- *Gain a Tangible*: i.e. the purpose of behaviors aiming at having access to materials or preferred stimuli;
- *Escape*: i.e. the purpose of behaviors aiming at avoiding unpleasant situations, such as attention, demanding tasks, or boring activities.

³<https://breakthru.com.my/fba/>

⁴<https://kindbh.com/talk-the-talk-whats-the-function-tangible-access-and-sensory-automatic-functions-of-behavior/>

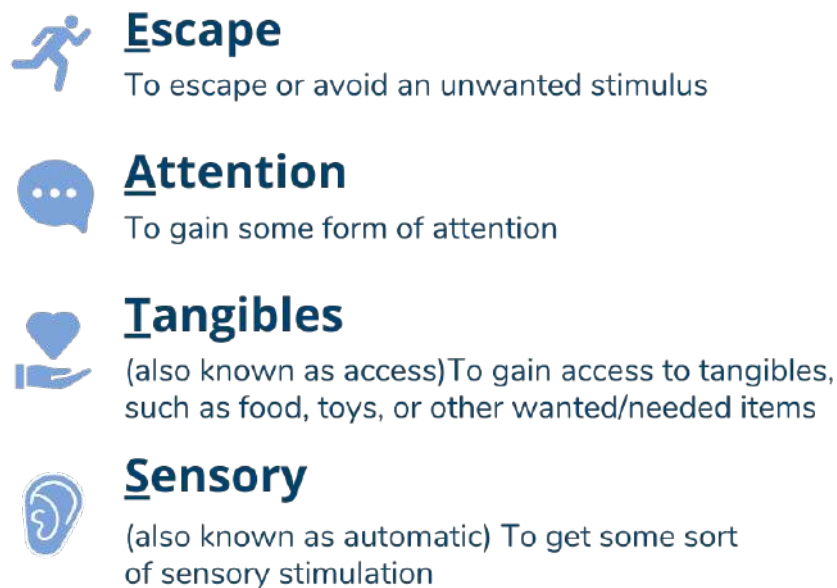


Figure 2.7 Four functions of behavior.⁴

ABA assessment methods are widely implemented in schools (40), where FBA offers educators a structured approach to identifying and addressing the root causes of challenging behaviors. By using FBA, teachers are better equipped to manage complex student behaviors in a more targeted and effective manner. On the other hand, robots as tutors have the potential to provide highly individualized attention, focusing on a single child's learning needs, thereby complementing traditional classroom strategies and supporting personalized learning experiences.

ABA is highly relevant to robotics, particularly in programming robots to manage and modify behaviors in educational settings. (41) demonstrated the effectiveness of a social robot, NAO, for social skills training in children with Autism Spectrum Disorder. The robot mediated interactions between the child and the instructor, improving eye contact and verbal initiation, with visual analyses confirming its positive impact. Similarly, (42) reviewed evidence-based strategies for reducing disruptive classroom behaviors. By integrating ABA principles, both robots and teachers can employ targeted interventions to promote desired behaviors, providing individualized attention and support in managing behavior and enhancing learning outcomes.

2.6 Planning in HRI

A plan can be understood as a representation of future behavior, typically comprising a set of actions with temporal and other constraints, intended for execution by an agent or multiple agents. PDDL, which stands for Planning Domain Definition Language, is a standard language used to encode classical planning tasks. Some of its key components include:

- *Objects*: Entities in the world that are of interest;
- *Predicates*: Properties or relationships between objects, which can be either True or False;
- *Initial State*: The state of the world before any actions are executed;
- *Goal Specification*: The desired conditions or outcomes that need to be achieved (i.e., what should be True);
- *Actions/Operator*: Methods for altering the state of the world.

It is important to note that a planning task specified in PDDL is divided into two parts: on one hand the domain, essentially describes everything that could occur in the world. On the other hand, the problem defining the current state of the world and the objectives to be achieved.

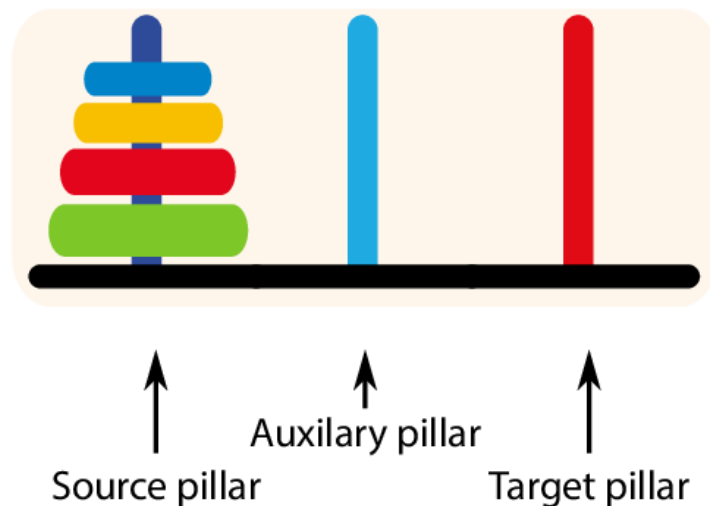


Figure 2.8 Hanoi tower (3).

One of the most famous examples used to introduce PDDL is the Tower of Hanoi. This classic puzzle, which involves moving a set of disks from one peg to another while adhering

to specific rules, serves as an ideal illustration of how PDDL can be used to model planning tasks. By defining the objects (disks and pegs), specifying the initial state (all disks on the first peg), and setting the goal (all disks moved to the target peg), PDDL enables a planning system to find the sequence of actions necessary to solve the puzzle, ensuring that no larger disk is ever placed on a smaller one.

Obviously, this is not the only way PDDL is employed, but it provides a solid introduction to its potential applications. As (43) highlights, a crucial task in developing effective robotic applications is the design of complex behaviors. The creation or automatic generation of a plan plays a key role in achieving this goal. The latter approach, in particular, is more desirable as it automates a task that would otherwise require considerable effort, making the process more efficient and scalable.

In the field of robotics, the deployment of robots in public and populated environments is gaining increasing attention. This creates a demand for more sophisticated planning techniques, particularly for service robots, which must navigate complex scenarios such as moving through public spaces. Moreover, techniques studied even by social robotics are employed, for instance, they help robots determine how to pass near moving people. Various planning algorithms are employed to address these challenges. PDDL has also played a crucial role in enhancing collaboration between humans and robots, both in public environments and industrial settings (44) (45). A similar interesting example is the work of (46), which employs a PDDL planner in dynamic environments to find paths that overcome obstacles.

Chapter 3

Methodology

3.1 General Architecture Overview

The architecture depicted in Figure 3.1 is designed to facilitate interaction between the robot Pepper and a child during a memory game. The system enables the robot to guide the child through the game while addressing any challenges or problematic behaviors that might arise. By integrating real-time behavior monitoring, adaptive planning, and natural language understanding, the robot responds dynamically to the child's actions and emotional state.

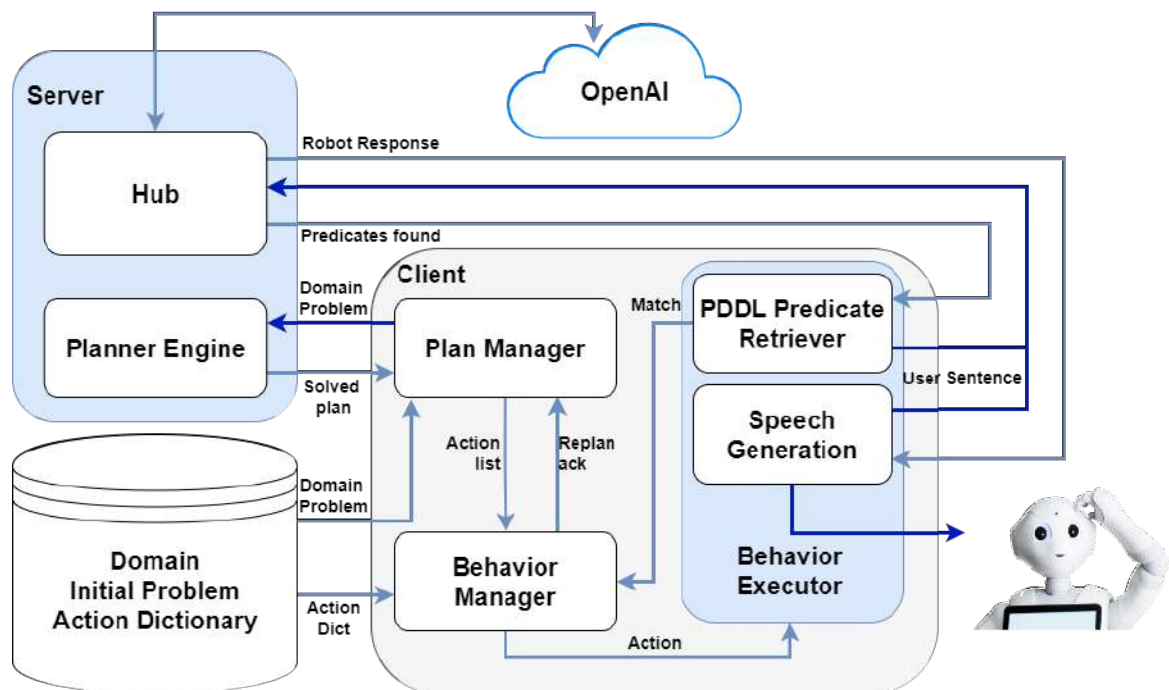


Figure 3.1 Architecture overview.

To accomplish this, the architecture leverages the robot’s built-in sensors and integrates external services like OpenAI to generate context-aware speech and gather valuable information from the conversation, helping the system assess the child’s emotional state.

On the server side, the architecture consists of two key components: the Hub and the Planner Engine. The Hub handles communication with external services like OpenAI, which enables the robot to generate context-aware speech and analyze the conversation to assess the child’s emotional state; this connection ensures the robot’s responses are personalized. The Planner Engine processes the PDDL domain and problem files received from the robot. Using the Fast Downward automated planner, the Planner Engine generates a sequence of actions that the robot will use to guide its interaction with the child. In this scenario, the actions might include explaining the game rules, providing feedback, or reacting to the child’s behavior.

On the client side, which is embedded in the robot, the system uses a set of components to manage and execute the plan. The Plan Manager serves as the central controller, responsible for sending the domain and problem files to the Planner Engine and retrieving the generated plan. It ensures the plan is executed in accordance with the child’s behavior and manages replanning when necessary. If the child’s actions deviate from the expected behavior, the Plan Manager may trigger a replan to adjust the robot’s actions in real-time.

Once a plan is retrieved, the Behavior Manager supervises its execution. This component ensures each step of the plan is carried out by verifying that the preconditions for an action are met before it is executed. As actions are performed, the system uses the PDDL Predicate Retriever to assert new predicates based on the robot’s interaction with the child and the game environment. These predicates are crucial for aligning the robot’s behavior with the current game state.

Speech generation plays a critical role in this system, allowing the robot to communicate with the child naturally. As the child speaks, their words are captured and transcribed using external Microsoft services. The transcribed text is then analyzed, enabling the robot to provide context-appropriate responses. The Speech Generation module ensures, in fact, that the robot’s verbal responses are coherent and aligned with both the current state of the game and the child’s emotional or behavioral cues.

The robot employs a *match and go* approach, where the Behavior Manager continuously evaluates the game environment by comparing the asserted predicates with the preconditions and effects of actions in the plan. If the conditions for the next action are met, the robot proceeds with execution. If a mismatch occurs — such as when the child’s response does not align with the expected behavior — the Behavior Manager signals the Plan Manager

to initiate a replan. This adaptability allows the robot to remain responsive and ensures a seamless interaction throughout the game.

3.2 Implementation of PDDL

With a comprehensive understanding of the psychological approaches outlined in Section 2.5, a PDDL domain has been developed to autonomously manage various scenarios of interest in accordance with the ABA approach. This framework is designed to be adaptable, allowing it to respond to diverse situations and make real-time adjustments based on changing conditions.

Specifically, the system operates by uploading a domain file, which describes the world or environment, along with an initial problem file that represents the current situation faced by the user. These files are then sent to the server where the planning engine is hosted, which then generates an optimal sequence of actions or a *plan*. Once the plan is created, it is sent back to the client, where the robot executes the actions during its interaction with the child.

Throughout this interaction, the robot is capable of identifying new predicates, conditions or changes in the environment by using OpenAI services to analyze the situation in real-time. If these new predicates do not align with the expected conditions for a consistent period, the system triggers a replanning process to ensure that the robot's actions remain relevant and effective.

3.2.1 Planner Engine

The planner engine selected for this work is Fast Downward (FD), which offers an efficient approach to addressing complex planning problems. Its innovative combination of multi-valued task representations, causal graph heuristics, advanced PDDL feature support, and a variety of search optimizations make it a versatile tool for both academic research and practical applications (4).

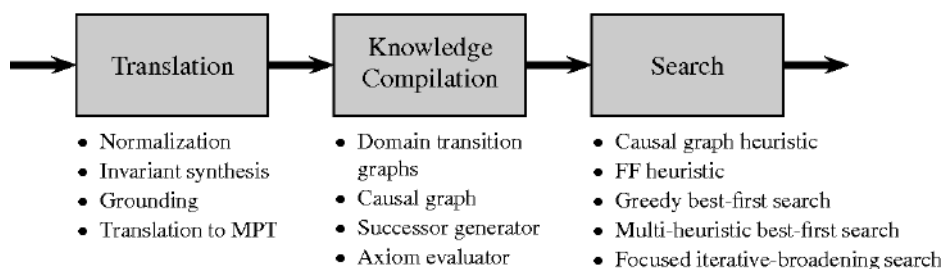


Figure 3.2 Fast Downward phases (4).

FD, in fact, excels at solving deterministic planning tasks by transforming PDDL problems into multi-valued representations, which reveal implicit constraints and improve efficiency in processing, especially in larger and more complex domains. This unique feature enables FD to scale effectively, where other planners might struggle. Figure 3.2 illustrates the procedure used by FD to solve planning tasks, providing a clearer understanding of its approach and architecture (4).

One of the defining features of FD is its causal graph heuristic, which proceeds “downward” in so far as it tries to solve hierarchically planning tasks. Starting from top-level goals, the algorithm recurses further and further down the causal graph until all remaining subproblems are basic graph search tasks. This enables FD to perform well in domains where intricate interactions between actions must be carefully considered.

The command used to run the planner engine is:

```
./fast-downward.py domain.pddl problem.pddl -evaluator h=ff() -search  
astar(lmcut())
```

The first argument identifies the planner engine, followed by the domain and the problem. This setup uses A* search, which guarantees finding an optimal solution, making it ideal for domains where cost-effectiveness is crucial, like the one presented in this project. Moreover, the LM-Cut heuristic ensures that the estimated cost to reach the goal never exceeds the actual cost, providing precise guidance by identifying and cutting essential steps in the problem without sacrificing optimality. The Fast Forward heuristic, specified with `h=ff()`, offers a quick estimate of goal distance by ignoring negative interactions between actions. While not always optimal, it is highly efficient and accelerates the search process, a really important feature since the necessity of planning in real-time during the HRI. By combining this fast heuristic evaluation with an optimal search strategy FD improves overall search speed without compromising solution quality.

3.2.2 Theory of Mind

Thus far, the modeling of verbal actions and concrete objects in PDDL has been relatively uncomplicated. However, a significant challenge arises when representing more abstract concepts, such as the user’s mental state.

Before implementing strategies to address *challenging* behaviors, it is crucial to gain a deeper understanding of the child’s mental state, specifically their goals, beliefs, and intentions. This is achieved through the action `DiscoverToM`, which initiates a conversation

with the child to infer their mental state (see Section 3.3). Inspired by the work of (16), this action is modeled using various predicates to accurately capture the child's ToM.

This can be better illustrated with an example. As soon as the action *DiscoverToM* is triggered, through dialogue facilitated by OpenAI services, the robot investigates the user's beliefs. When evidence is found, the robot updates its understanding of the user's mental state and intentions. For instance, consider the predicate `(task_is_hard)`. The subsequent action `ConfirmHardTask` clarifies the ToM of the child by using three specific predicates:

- `(is_hard ?i1 ?t)`: This predicate identifies that task `t` is hard, with identifier `i1`.
- `(belief ?i2 ?a ?i1)`: This predicate indicates that agent `a` holds the belief represented by `i1`, meaning the child believes that the task is hard.
- `(is_true ?i2)`: This predicate signifies that, according to the robot, the belief `i2` (the child believes the task is hard) is true.

Although this theorization may seem relatively simple, it provides the robot with a clear representation of the user's mental state. Without such a mechanism, it would not be possible to ensure the application of the correct strategy to address the user's needs effectively.

3.2.3 Behavior Functions and Strategies

As stated in Section 2.5, drawing from ABA and FBA principles, two behavior functions were identified for testing the model's validity: *Gain a Tangible* and *Escape*. The aim is that, through verbal and non-verbal interaction with the child, the robot would be able to autonomously recognize the occurrence of the following behaviors:

- *Gain a Tangible*: When pleasant activities are interrupted by time limits, negative reactions of the child may be elicited to prolong the activity;
- *Escape*: When presented with a task, a child may perceive it as too difficult or too boring, potentially provoking adverse responses to avoid the task.

With a deep understanding of these psychological approaches, it has been developed a PDDL domain capable of autonomously managing various relevant scenarios where the two behavior functions mentioned above can influence a child's behavior. Table 3.1 details the actions that have been modeled, omitting parameters, preconditions, and effects for the sake of brevity. All these details and more can be found in Appendix A.

Action	Description
ConfirmBoringTask	Confirm <i>Escape</i> function
ConfirmHardTask	Confirm <i>Escape</i> function
ConfirmWant2Play1	Confirm <i>Gain a Tangible</i> function
ConfirmWant2Play2	Confirm <i>Gain a Tangible</i> function
DiscoverTOM	Prompt dialogue to assess child's ToM
GoodJob	Praise child for achievement
GoodJobStrategy	Praise child for achievement with strategy
Interact	Engaging conversation with the child
PresentTask	Explain task rules in detail
PutAway	Ensure the child stops playing
StrategyBoringTask	Apply strategy if child finds task boring
StrategyHardTask	Apply strategy if child finds task hard
StrategyWant2Play	Apply strategy if child still wants to play
Wait	Wait and monitor task execution
WarningCleanUp	Warn about time and monitor completion

Table 3.1 PDDL actions related to behavior management.

The list of available PDDL actions includes those that enable the robot to apply the appropriate intervention strategy once a behavioral function has been detected.

In particular, these functions have been formally represented in PDDL through actions such as `ConfirmHardTask` or `ConfirmBoringTask` for the *Escape* function, and `ConfirmWant2Play1` or `ConfirmWant2Play2` for the *Gain a Tangible* function. The predicates providing the precondition for the action to be selected are detailed in Table 3.2.

Function	Action	Predicates	Strategy
<i>Gain a Tangible</i>	ConfirmWant2Play1	want_to_play	StrategyWant2Play
	ConfirmWant2Play2	touching	
<i>Escape</i>	ConfirmHardTask	task_is_hard	StrategyHardTask
	ConfirmBoringTask	task_is_boring	StrategyBoringTask

Table 3.2 PDDL modeled behavioral functions and corresponding strategies.

To illustrate the complete planning process more clearly, consider the following scenario. The initial problem generates a plan aimed at proposing a task and monitoring its advancement:

```
(Interact game kid memory)
(PresentTask game kid memory)
(Wait game kid memory)
(WarningCleanUp game kid memory)
(PutAway game kid memory)
(ConfirmAllGood game kid memory x1 x2)
(GoodJob game kid memory x1 x2)
```

In particular, the initial plan proposed involves a robot presenting a memory card game to a child. The process begins with a brief interaction (`Interact ...`), followed by an explanation of the rules (`PresentTask ...`). The robot then monitors the child while they play (`Wait ...`). As the time nears its end, the robot warns the child (`WarningCleanUp ...`), instructs them to stop and put away the cards (`PutAway ...`), confirms they agreed to finish (`ConfirmAllGood ...`), and ends with praise (`GoodJob ...`).

During the action (`Wait ...`), the robot interacts with children through OpenAI-supported verbal exchanges. When the robot detects a shift in the child’s engagement, prompts replanning starting from the current state (see Section 3.1), with the asserted predicate `not_doing` acting as a trigger. Since this predicate does not match the preconditions of the `WarningCleanUp` action, which was originally expected as the next step, but matches the preconditions of the alternative action `DiscoverToM1`, it signals the need for replanning. In response to this interaction, a new plan is formulated, which includes the following action:

```
(DiscoverTOM1 game kid memory)
```

Once again, this action involves verbal interaction with the child using OpenAI functionalities, aiming to ground predicates from a given list of possible options based on the child’s responses, which trigger replanning. A possible outcome is as follows:

```
(ConfirmHardTask game kid memory x1 x2)
(StrategyHardTask game kid memory x1 x2)
(Wait game kid memory)
(WarningCleanUp game kid memory)
(PutAway game kid memory)
(ConfirmAllGood game kid memory x1 x2)
(GoodJob game kid memory x1 x2)
```

This update, performed by the PDDL Predicate Retriever, leads to the conclusion that the task is perceived as too challenging for the child, once again prompting the Plan Manager to initiate a replanning process. In this specific case, the asserted predicates match the preconditions of the `ConfirmHardTask` action, indicating the behavioral function *Escape*, which necessitates the specific strategy `StrategyHardTask` (see Table 3.2).

The `ConfirmHardTask` action does not result in a visible change in the interaction or the environment, as its primary function is to validate the situation rather than drive the robot's behavior. The key intervention occurs with the `StrategyHardTask` action. Specifically, the robot suggests using fewer cards to reduce potential confusion and discomfort for the child.

However, this is just one possible scenario. After the strategy has been applied, other behavior functions might occur. For instance, if the child disagrees with ending the task when the time is up, a new plan will be generated. In this second case, after the guess about the behavior function *Gain a Tangible* has been confirmed, the planning result would be the following:

```
(ConfirmWant2Play1 game kid memory x1 x2)
(StrategyWant2Play game kid memory x1 x2)
(GoodJobStrategy game kid memory x1 x2)
```

In this plan, it is first confirmed that the child still wants to play with the memory game, and the *Gain a Tangible* strategy is therefore applied. Precisely, the robot will grant an additional two minutes of play with a timer, and then it will ask the child to help clean up the cards.

The overview in Figure 3.3 presents the overall planning scheme, while a detailed explanation of each action can be found in Appendix A. However, it is important to note that this graph represents only a subset of possible plans and does not encompass all potential outcomes.

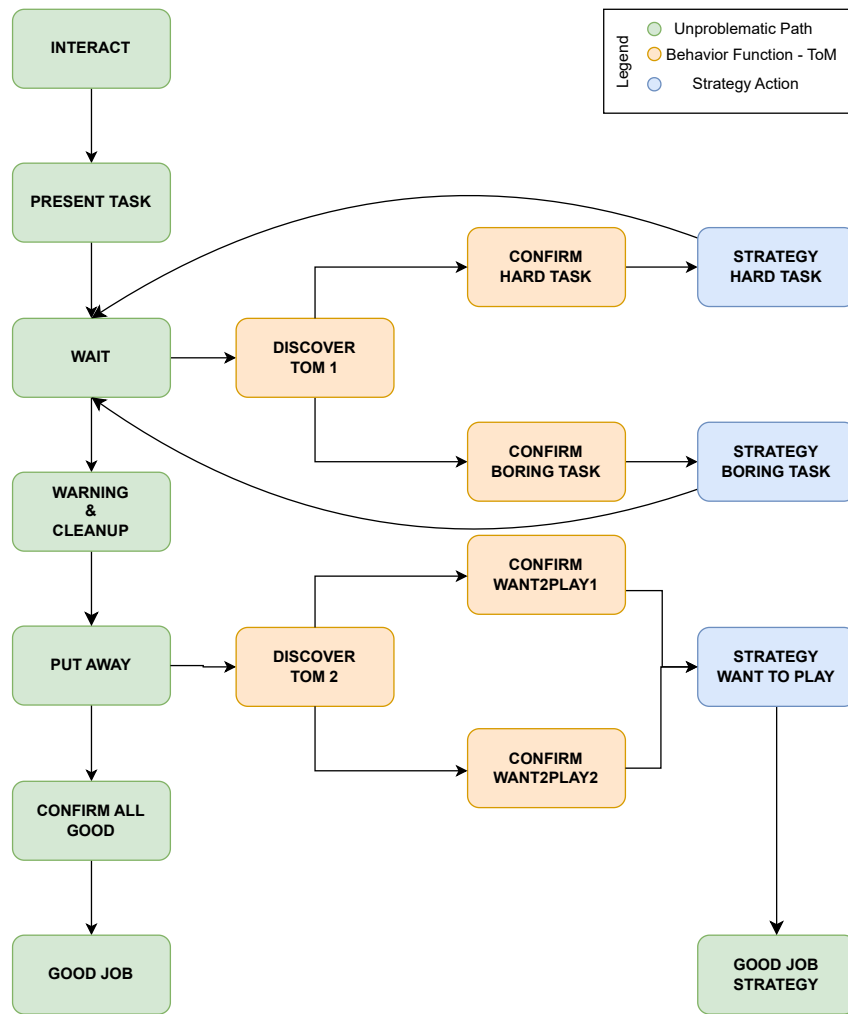


Figure 3.3 Planner overview.

3.3 Action Implementation

The actions designed in the PDDL framework must then be implemented, though each one is implemented differently based on its role in the system. These actions are grouped into three main categories, as outlined below.

3.3.1 Actions that do not require execution

These actions do not result in a visible change in the interaction or the environment, as their primary function is to validate the situation by clarifying the robot's belief about the user's mental state, rather than driving the robot's behavior.

Action	Description
ConfirmBoringTask	Confirm <i>Escape</i> function
ConfirmHardTask	Confirm <i>Escape</i> function
ConfirmWant2Play1	Confirm <i>Gain a Tangible</i> function
ConfirmWant2Play2	Confirm <i>Gain a Tangible</i> function

Table 3.3 PDDL actions that do not require execution.

3.3.2 Actions that can be scripted

This category is the first to practically be implemented, with each behavior developed and installed on the robot. By scripting these actions, the robot can execute them with consistent timing, content, and style across multiple trials and interactions. This consistency is crucial for ensuring a predictable and reliable user experience.

The actions hosted in this category can then be grouped into several subcategories.

Action	Description
StrategyBoringTask	Apply strategy if child finds task boring
StrategyHardTask	Apply strategy if child finds task hard
StrategyWant2Play	Apply strategy if child still wants to play

Table 3.4 PDDL strategy actions that can be scripted.

These strategies involve predefined responses and interventions based on certain conditions or inputs, ensuring that the robot's reactions are consistent and reliable during interactions. The script and the strategies are chosen according to the ABA theories (8)(47);

Action	Description
PresentTask	Explain task rules in detail

Table 3.5 PDDL presenting action that can be scripted.

This behavior involves the structured presentation of tasks to users, including clear instructions and prompts, to ensure that tasks are delivered consistently across different interactions;

Action	Description
GoodJob	Praise child for achievement
GoodJobStrategy	Praise child for achievement with strategy

Table 3.6 PDDL praising actions that can be scripted.

These actions involve evaluating and providing feedback on the completion of a game or activity. They ensure that the feedback is given uniformly, regardless of the specific game or activity;

Action	Description
WarningCleanUp	Warn about time and monitor completion

Table 3.7 PDDL warning action that can be scripted.

To prepare users for the conclusion of a game or activity, this scripted behavior issues a timed warning, maintaining consistency in how and when users are notified before the game ends.

3.3.3 Interactive actions

Action	Description
DiscoverTOM	Prompt dialogue to assess child's ToM
Interact	Engaging conversation with the child
PutAway	Ensure the child stops playing
Wait	Wait and monitor task execution

Table 3.8 PDDL interactive actions.

As described in Section 3.2 regarding the ToM recognition mechanism, these actions enable direct interaction between the humanoid robot and the user by leveraging OpenAI's Large Language Models (LLMs) through a custom-designed prompt. This prompt allows the robot to continue the conversation by responding to the user's statements and ground relevant predicates from the interaction. Both tasks, which are based on the same user sentence, are now discussed.

It is important to note that all actions in this category are aimed at the same overall goal: interacting with the user. However, each action has a subtle variation in its conversational focus. For instance, `DiscoverToM` generates sentences more inclined toward investigating the user's mental states, while `Wait` is more focused on checking whether the child is progressing in the game and remaining engaged. The key to achieving this distinction lies in the prompt design, which tailors the conversation based on the specific action, ensuring that the generated sentences align with the intended purpose.

On the other hand, predicate grounding is accomplished through a carefully crafted prompt that explicitly queries OpenAI to ground predicates based on the child's responses. The list of predicates to be grounded through conversation includes those related to both the child's activities and mental state regarding the task. This serves as the foundation for identifying task-related issues by directly engaging with the child.

Given the importance of prompt design, it is described in detail in the following paragraphs.

3.3.4 Prompt Design

To obtain two different responses from the LLMs, two distinct prompts are used, based on the same user sentence but supplemented with different additional information. The prompts

sent to OpenAI have been designed, after multiple trials, based on the work of (48). They are constructed to maximize the effectiveness of the responses. The following considerations are taken into account for its design:

- *Context-awareness*: The system includes relevant speaker information, allowing OpenAI to generate responses that are aligned with the current situation;
- *Task-specific instructions*: Prompts are structured to include clear and specific instructions, helping the model focus on relevant predicates or actions needed;
- *Predicate grounding*: The server incorporates the user's command into the prompt to align with the predefined predicates, allowing the OpenAI model to perform logical operations or return valid responses based on available data.

An example of a prompt for predicate selection is now reported. Given the following information:

User Sentence:

```
"Uff, questo gioco è troppo difficile. Non ci voglio più giocare"
```

Possible Predicates:

```
["is_hard", "is_boring", "doing", "none"]
```

The resulting system prompt shall be:

```
You have to choose ONE OR MORE predicates from a list that best fits the sentence provided by the user.
```

```
The list will be delimited by square brackets, while the sentence will be delimited by ###.
```

```
Reply only with the name(s) of the predicate(s), eventually preceded by a "not" (only if needed), separated by commas if there are multiple.
```

```
If no predicate in the list seems to be appropriate, the predicate should be 'none' (in lowercase).
```

```
[is_easy, is_difficult, doing, completed, none]
```

```
###Uff, questo gioco è troppo difficile. Non ci voglio più giocare###
```

This method ensures that the model can focus solely on matching the user's sentence to the available predicates. At the same time though, the system is asked to generate a response that creates a realistic social interaction prompt. From the speaker information uploaded to the server, the following details are known:

Speaker Info: Giulia, female, 8

For this request, the system prompt would be constructed as follows:

Pretend to be a social robot named Pepper. You can feel emotions and you have to express preferences. You are talking with Giulia. You have to reply to Giulia in Italian, with just ONE brief sentence. Explicitly mention Giulia's name and do not greet.

Reply accordingly to:

"Uff, questo gioco è troppo difficile. Non ci voglio più giocare"

You need to check if Giulia is doing the task assigned.

To improve the sentence generation,

you also have the following information at your disposal, each surrounded by two hashtags.

#Optional: Giulia is a female and is 8 years old.#

#Compulsory:

Do not use emojis and try to entertain the person you are talking with.#

3.4 Server-Client Model

The implemented system follows a server-client architecture designed to facilitate task planning and execution for the Pepper robot.

The client, which operates directly on the robot, is responsible for interacting with the robot's sensors, motors, the server, and speech recognition modules. On the other hand, the server handles complex planning tasks and speech processing. This architecture enables the robot to offload computationally intensive tasks, such as PDDL plan generation and natural language understanding, to the server.

The client-server communication is established through RESTful API methods.

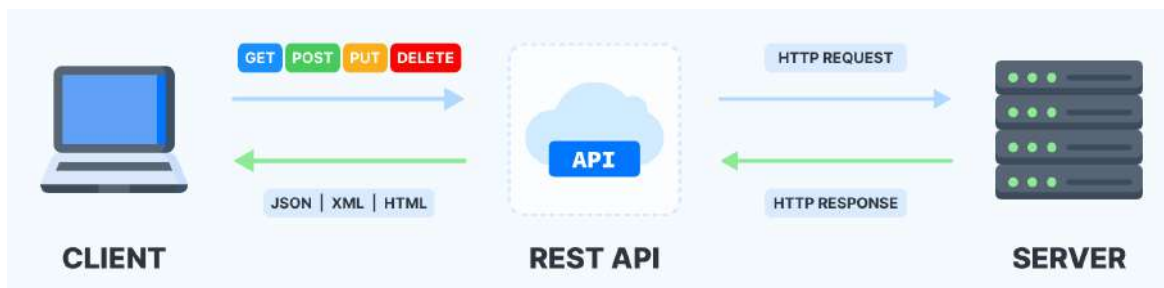


Figure 3.4 RestAPI model.¹

In the proposed architecture, the client sends requests and the server responds. As represented in Figure 3.4 REST is an architectural style for distributed systems, following a client-server model. Requests and responses involve exchanging *representations* of resources, such as data identified by URIs. REST is based on three key principles:

- *Client-Server Separation*: Clients handle the interface, and servers manage data, allowing independent development;
- *Stateless Communication*: Each request is self-contained, with no server-side context;
- *Uniform Interface*: Standardized operations like Create, Retrieve, Update, and Delete ensure consistent interaction.

This design promotes simplicity, scalability, and flexibility. The client sends requests to the server, which processes these requests, performs the required tasks, and responds with the relevant data.

3.4.1 Uploads

Inside the robot, the client application contains a directory named *uploads*, which houses the following files:

```
actions.json
domain.pddl
problem.pddl
```

Among these, the `domain.pddl` and `problem.pddl` files were discussed in detail in Section 3.1, while a comprehensive explanation of the predicates and actions is provided in Appendix A. The JSON file is particularly relevant since outlines various actions that the robot can undertake, specifying the conditions that must be met before an action can be performed and the outcomes resulting from the action. Additionally, the file includes time estimates for each action, ensuring that the robot operates within expected time frames.

Each action in the JSON file has the following attributes:

¹<https://gotapi.com/what-is-restful-api/>

- *Preconditions*: Conditions that must be true for the action to be executed. These represent the current state of the robot or the environment;
- *Effects*: Changes of the state or outcomes resulting from performing the action. These indicate how the action will modify the robot's state or environment;
- *Min* and *Max*: Time constraints for how long the action will take to complete, expressed in seconds. This helps manage the timing of operations to ensure they fit within expected durations.

This is crucial for several reasons. First, it provides flexibility and adaptability: whenever a new scenario needs to be addressed, it is possible to simply replace the domain and problem files within the application. This makes it easy to update and adapt to new planning tasks without altering the core system. Similarly, the JSON file, which defines actions and their parameters, can be updated in the same way. This modular approach ensures that the robot can be quickly reconfigured to handle different tasks or scenarios by simply modifying the contents of the directory.

3.4.2 Client

As mentioned earlier, the client is embedded within the robot. The specific robot used for the application described is Pepper (see Fig. 3.5), a humanoid robot developed by SoftBank Robotics. Pepper is designed to interact with humans through conversation and touch. Equipped with various sensors, cameras, and microphones, it can even recognize faces.

The main software used to program the Pepper robot is Choregraphe, which allows for the creation and customization of behaviors. The client proposed is a Python Class script embedded within a Choregraphe behavior.

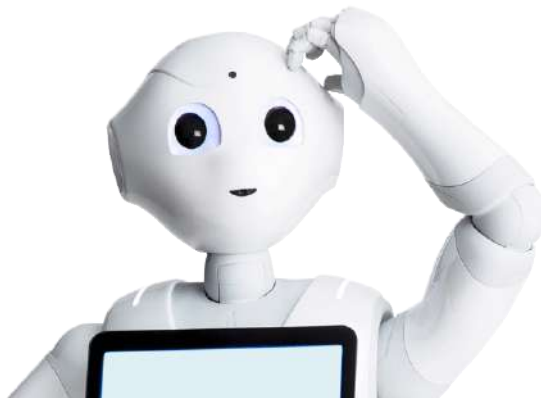


Figure 3.5 Pepper robot.²

After initializing all necessary components and handling the lifecycle methods (`onLoad()`, `onUnload()`, `onInput_onStop()`), the class provides several key functionalities. Here is a detailed explanation of each method:

- `publish_file(file_path):`
This method uploads a specified file (`domain.pddl` or `problem.pddl`) to the server. It reads the file from the given path, prepares it for upload, and sends a POST request to the appropriate server endpoint. After sending the file, it checks the server's response and logs whether the file was published successfully or if there was an error.
- `get_plan():`
This method retrieves a plan from the server. It sends a GET request to the server and processes the response to extract the plan. The plan, which is initially a string, is converted into a list of actions. If the retrieval or conversion fails, it logs an error.
- `predicate_list():`
This method sends a list of predicates to the server. It parses the domain PDDL file to create a list of predicates and then attempts to POST this list to the server. If the request fails, it retries and logs the result of the operation.
- `replan(new_predicates):`
This method updates the problem PDDL file with new predicates and tries to get a new plan. It parses the current domain and problem files, updates the problem file with the new predicates, and writes the updated file. The method then publishes the updated problem file, retrieves a new plan from the server, and executes it if available. If no plan is found, it logs an error.
- `execute_plan(plan):`
This method executes a list of actions specified in the plan. It iterates through each action, determines its type, and either executes it directly, uses the `ActionManager` class to start another behavior, or skips it based on predefined lists (see 3.5). The method logs the status of each action and continues until all actions are executed.
- `execute_action(act_name):`
This method executes a specific action based on its name. It retrieves details about the action, including preconditions, effects, and maximum and minimum durations. The method resets relevant variables and manually adjusts predicates. It then continuously

²<https://www.directindustry.it/prod/aldebaran/product-177224-2477756.html>

checks if the action's effects are achieved or if re-planning is needed. It uses the microphone to capture and process audio input, it updates predicates and replans if necessary.

- `mic()`:
This method captures the audio input and extracts a message from it.
- `openai(current_action)`:
This method processes the transcribed audio message and updates the list of predicates using the LLMs provided by OpenAI. It prepares a JSON payload with the transcribed audio message and current action and sends it to the server for processing. The method handles the server's response by updating the answer and `actual_predicates` based on the processed data.

Finally, `onInput_onStart()` method handles the overall logic when starting the process. It checks if the `domain.pddl` and `problem.pddl` files exist. It loads the actions dictionary from a JSON file, publishes the domain and problem files to the server, and sends the list of predicates. It retrieves a plan from the server and, if a valid plan is obtained, executes it.

To improve the functionality of the client, additional classes aimed at supporting the principal client class have been implemented. Firstly, a class named `StringChecker` has been developed, which is equipped with tools for handling strings and performing checks on data. This class is fundamentally a tool for managing and verifying strings and data conditions, ensuring that lists and dictionaries are aligned according to specified rules.

In addition, the `Parser` class implemented provides comprehensive tools for parsing PDDL files, constructing problem descriptions, and manipulating predicates and objects. It ensures that PDDL data is correctly formatted and can be used effectively in planning systems.

Lastly, the `ActionManager` class is a practical utility for managing and executing various actions on a robot. It ensures that each action is performed only if the necessary behavior is installed on the robot. Before executing an action, the class verifies the presence of the required behavior, preventing errors due to missing functionalities.

3.4.3 Server

The server is developed using the Flask web framework, providing a robust solution for handling various requests related to domain and problem planning. The server is composed of two main parts: on one hand there is the *Planner Server*, which handles file uploads for

domain and problem files and generates plans based on these inputs. On the other hand, the *OpenAI Server* processes predicate lists for natural language understanding and transcribed audio inputs for speech generation using OpenAI API.

Here the methods implemented are reported and explained:

- `upload_file(file_part, file_type):`
This helper function handles file uploads, saving the file to a specified directory. It logs the upload time, duration, and updates the `upload_times` dictionary. If the upload fails, it logs an error. In particular, it is used to upload the domain or the problem file
- `get_plan():`
This route generates a plan using the Fast Downward planner. It verifies the existence of both the domain and problem files. It then runs the planner as a subprocess and logs the result. If successful, it reads the plan from the `sas_plan` file and logs the time taken for planning. If there is any failure, it logs the error. The route returns the generated plan or an error message depending on the outcome.
- `upload_predicate_list():`
This route accepts JSON data containing a predicate list, stores it in the global variable, and logs the successful receipt of the data. It returns a success message in response.
- `process_audio():`
This route processes audio input. It accepts the transcribed user audio and the `current_action`. It uses the OpenAI methods class to retrieve the user answer message along with the updated predicate list generated.

Both Azure services and the OpenAI API play key roles in the system's architecture, and their functions will be described in detail in the following paragraphs.

Azure

Azure plays a crucial role in the server for speech transcription tasks. Specifically, the system leverages *Azure Cognitive Services*, particularly the *Azure Speech-to-Text API*, to convert audio inputs into text. This transcription process forms the foundation for further natural language understanding tasks, such as predicate grounding and planning operations.

The Azure Speech Service ensures accurate and efficient transcription, supporting multiple languages and speaker variations. This makes it an ideal choice for processing diverse audio inputs that are passed to the server. Once the audio is transcribed, the resulting text is

further analyzed using OpenAI API for generating responses or extracting key predicates from the speech.

OpenAI

As stated in Section 3.3 the Server exploits OpenAI powerful models to process and analyze the text generated from Azure speech transcription. OpenAI is used for natural language understanding, predicate grounding, and generating responses based on the user's commands. By utilizing OpenAI, the server can handle complex conversational inputs and provide meaningful, context-aware responses.

The server employs OpenAI GPT-4o and GPT-3.5 turbo models to interpret the transcribed text. Once audio is converted into text via Azure, the OpenAI API is called to process the text and identify key predicates or perform specific actions based on the user's instructions. This process is tailored by considering the current action and stored predicate list, allowing the system to understand user intent and provide relevant outputs.

3.5 Implementation Process

To build this framework, several steps are involved. It begins with a deep analysis and modelization of the PDDL framework. This involves carefully considering the causality of actions and determining the most appropriate predicates. The goal is to ensure that the predicates not only maintain their meaning and can be effectively recognized by OpenAI but also facilitate the correct execution of actions at the right moments. For a detailed list of actions and predicates, along with brief explanations of their meanings, refer to Appendix A. Then the actions that can be scripted are the first to be implemented and tested on the real robot.

Once the PDDL and the other behaviors are ready, the focus shifts to the server side. This involves defining the main functionalities and key exchanges between the client and the server. Specifically, the primary data exchanges considered are the uploading of domain and problem files and the retrieval of the plan. This part, referred to as the *Planner Server* in Section 3.4, is implemented first as the server-side component of the REST API. The mechanisms for checking the planning engine and transferring the documents are thoroughly reviewed and tested.

The client is developed by first spending a significant amount of time creating pseudocode and conceptualizing the underlying mechanism, which can be summarized as follows.

Generally, the system operates in a cyclical manner, where the client and server engage in continuous data exchange to facilitate task execution and dynamic replanning. Initially, the client uploads the PDDL domain and problem files to the server, which processes them using the FD planner, as described in Section 3.1. The server then generates and returns a plan, which the client follows. During task execution, the client monitors the state of the environment and, when verbally interacting with the user, sends the transcribed user audio to the server for speech generation and grounding using OpenAI models. Predicates describing the current situation are generated through the models. If they indicate a deviation from expected conditions, the client requests a new plan from the server

It is important to explain how the mechanism of predicate retrieval works. It would not make sense for OpenAI to update all possible predicates every single time, as this would create an unnecessary workload and reduce accuracy. Here is shown how the process works instead: when the server loads the first domain, it is also given a list of relevant predicates, extracted by the client, and stores them in a global variable. However, the twist is that the list does not contain all possible predicates; it only includes those referring to the agent (see Appendix A). The reason behind this approach refers to the fact that most predicates concern the state of the robot, which is determined by the action itself, rather than by the dialogue or interaction. So, when an action begins, all predicates unrelated to the agent's state are already taken into account. OpenAI then retrieves a predicate if it is not already in the list, or, if it has already been mentioned, it will not be removed unless explicitly negated in the dialogue. If a predicate is negated, it will be inserted again in its negated form. If no predicates are recognized, none is added. For example, if a user mentions that is performing a task, the predicate `doing` is registered. In the following sentence, even if the user talks about something else, this does not imply that they has stopped doing the task, so the predicate is not removed or negated, this mechanism will be also clarified in Section 4.2.

This architecture ensures an adaptable and efficient task execution system, allowing the robot to offload complex planning and language processing tasks to the server while focusing on real-time interaction and action execution.

In conclusion, this framework effectively integrates multiple components to create an adaptive system for task execution. The use of the FD planner ensures efficient planning and problem-solving capabilities, while the OpenAI models provide advanced speech generation and proper context understanding. The cyclical data exchange between the client and server facilitates real-time interaction and dynamic replanning, making the system versatile and responsive to changing conditions.

Chapter 4

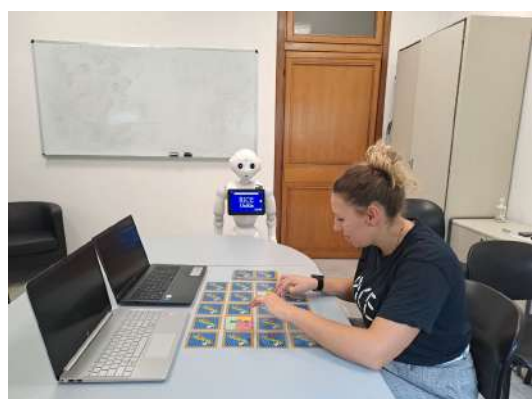
Framework Evaluation

4.1 Experimental Design

The experimental evaluation of the developed framework is conducted in a controlled laboratory setting focused on a memory game task, where participants match pairs of identical cards. The setup includes a one-on-one interaction between a human adult and the humanoid Pepper robot. Two laptops are used for this experiment: one serves as a server, while the other is used to run the client on the robot and store the necessary log information for the evaluation. The PC hosting the server also functions as an external microphone, as Pepper's built-in microphones cannot be used to acquire audio streaming for privacy reasons.



(a) Table setup.



(b) Pepper setup.

Figure 4.1 Experimental setup.

As shown in Figure 4.1, the environment is arranged with the participant and the robot positioned at opposite corners of a table. Firstly, the participant is given time to become familiar with interacting with the robot. After several minutes of casual interaction, the

experiment formally began, with the robot following a scripted introduction to present the task. Memory game cards, used as part of the experimental task, are pre-arranged in front of the participant. This is presented by Pepper, which shows the 20 cards placed face down on the table, as shown in Figure 4.2a. Additionally, two decks of cards, each containing 10 already shuffled couples, are positioned on the side. They are intended for the participant to use once the first round is completed, as explicated by the robot.

After the rules are explained by the robot and understood by the user, the latter starts to play while the robot monitors the advancement in the task given through verbal interaction. As visually represented in Section 3.2. The evaluation ends when the user completes the game by putting away all the cards.



(a) Memory disposition.



(b) Couples example.

Figure 4.2 Memory setup.

The session has been repeated multiple times with the same participant aiming to evaluate the robot's replanning abilities in controlled settings. This is done, as previously stated, by a one-on-one interaction between an adult pretending to be a child named Giulia, and the humanoid robot Pepper.

It should be noted that experiments with children are not the focus of this thesis. Instead, since collecting pre-trial data to assess the performance and risks of the proposed solution is a mandatory step before conducting experiments with robots interacting with people (also required by the Medical Device Regulation (EU) 2017/745), the remainder of this work focuses on the pre-trial assessment of system performance.

4.2 Measurements and Data Collection

The evaluation framework provides insights into the robot's performance in adapting to real-time disruptions during interactions, laying the foundation for future experiments with actual children. The key metrics assessed include:

- *Replan Time*:
Measures the time from the introduction of a disruption to the completion of the replan assessing the responsiveness of the replanning mechanism.
- *Cloud Response Time*:
Evaluates the time taken for the robot to send a request, process cloud information, and receive a response.

The qualitative evaluation includes assessing the *ABA implementation*, i.e., the robot's ability to ground predicates properly, assert the observed state of the child's activity and mental state, recognize behavior functions, and ultimately achieve intended outcomes despite interruptions.

The latter aspect, in particular, is assessed through a preliminary evaluation of the logs produced during one-on-one interactions between an adult participant and the robot. To better illustrate this evaluation and the overall framework, a commented version of the speech log is provided for both the implemented and non-implemented strategies. The process of retrieving predicates is detailed in Section 3.5.

No strategy implemented

Firstly, to validate the mechanism, the user is instructed to adhere strictly to the robot's instructions and avoid any problematic behavior. This controlled scenario is designed to prevent the triggering of the replanning process, allowing for an evaluation of the robot task execution without the need for adaptations.

The following plan is generated starting from the first domain uploaded in the robot.

```
[(interact game kid memory),  
(presenttask game kid memory),  
(wait game kid memory),  
(warningcleanup game kid memory),  
(putaway game kid memory),  
(confirmallgood game kid memory x1 x1),  
(goodjob game kid memory x1 x1)]
```

In this section is reported the verbal interaction log for the speech generated, regarding the actions defined as *Interactive* in Section 3.5. Note that in the **Log** column, user sentences are indicated by the letter U, the robot's responses by the letter R, and the retrieved predicates by the letter P. It is important to highlight that each action has a specified minimum and maximum time duration; this implies that even if the action performed achieves its intended effect (the predicates *match*), the action cannot be concluded until the minimum time has elapsed. The log provided corresponds to the first action of the generated plan: *Interact*.

Log	Translation	Commentary
U: Ciao Pepper, parliamo!	U: Hi Pepper, let's chat!	The interaction is initiated by the user, who starts the dialogue.
R: Mi piace molto parlare con te, Giulia!	R: I really like talking with you, Giulia!	
P: [not ready, start, none]	P: [not ready, start, none]	Note that the retrieved predicates correspond to those present at the start of the interaction, with the addition of none, since no further action or disruption has been detected at this point.
U: Anche a me piace molto parlare con te, di che cosa vuoi parlare?	U: I also really enjoy talking with you. What do you want to talk about?	
R: Giulia, parliamo dei tuoi giochi preferiti!	R: Giulia, let's talk about your favorite games!	It is important to highlight that the first predicate retrieved by the robot remains active throughout the interaction, as it was never negated and did not trigger a replan since it did not interfere with the current action.
P: [not ready, start, none]	P: [not ready, start, none]	
U: Mi piacciono tanti giochi...	U: I like many games...	Although the effects of the action are matched, the action itself does not conclude because the interaction is not strictly related to the game, and thus the predicates remain unaffected.
R: Giulia, anche io li adoro.	R: Giulia, I also love them.	
P: [not ready, start, none, want_to_play]	P: [not ready, start, none, want_to_play]	The effect has been reached, it is a match!
U: Allora potremmo giocare insieme	U: Then we could play together	
R: Sì, Giulia, giocare è divertente!	R: Yes, Giulia, playing is fun!	
P: [not ready, start, none, want_to_play]	P: [not ready, start, none, want_to_play]	
U: Quali sono i tuoi giochi preferiti?	U: What are your favorite games?	
R: Mi piacciono i giochi di costruzione, Giulia.	R: I like building games, Giulia.	
P: [not ready, start, none, want_to_play]	P: [not ready, start, none, want_to_play]	

The effect has been reached, it is a match!

When the effect is reached and the minimum duration time has elapsed, the robot proceeds to the following action of the generated plan, which is `PresentTask`. The rules are then explained and the game can start.

The next action log reported is: `Wait`. Here the Pepper robot interacts with the user to check if the task is running smoothly.

Log	Translation	Commentary
U: Possiamo parlare mentre gioco?	U: Can we talk while I play?	After the user inquires whether it is possible to maintain a conversation while playing, Pepper actively monitors the situation to ensure the user remains engaged.
R: Giulia, mentre giochi vuoi raccontarmi cosa stai facendo?	R: Giulia, while you play, do you want to tell me what you're doing?	
P: [warning, not waiting, doing]	P: [warning, not waiting, doing]	The predicate <code>want_to_play</code>
U: Questo gioco del memory mi piace tanto!	U: I really like this memory game!	is detected from the enthusiasm of the user during the play-time.
R: Giulia, sono felice che ti diverta!	R: Giulia, I'm glad you are having fun!	
P: [warning, not waiting, doing, <code>want_to_play</code>]	P: [warning, not waiting, doing, <code>want_to_play</code>]	
U: Tu hai mai giocato a memory?	U: Have you ever played memory?	A new predicate, <code>not task_is_hard</code> , has been
R: Giulia, adoro giocare e scoprire tutte le carte!	R: Giulia, I love playing and revealing all the cards!	added. However, this does not significantly affect the interaction or trigger a replan, as it remains aligned with the ongoing dialogue and the expected effect.
P: [warning, not waiting, doing, none, <code>want_to_play</code>]	P: [warning, not waiting, doing, none, <code>want_to_play</code>]	
U: Beh, ma è facile se scopri tutte le carte...	U: Well, it's easy if you reveal all the cards...	
R: Giulia, ma secondo te riusciresti senza scoprire tutte le carte?	R: Giulia, do you think you could do it without revealing all the cards?	
P: [warning, not waiting, doing, none, <code>want_to_play</code> , <code>not task_is_hard</code>]	P: [warning, not waiting, doing, none, <code>want_to_play</code> , <code>not task_is_hard</code>]	

Log	Translation	Commentary
U: Sì lo sto facendo	U: Yes, I'm doing it.	Pepper continues to maintain
R: Giulia, sono curioso di sapere come te la stai cavando!	R: Giulia, I'm curious to know how you're doing!	its interaction without needing to replan, as no disruption occurs. The action continues smoothly until the time is over. Only at this point, since the predicates still match, it is possible to move to the next action.
P: [warning, not waiting, doing, none, want_to_play, not task_is_hard]	P: [warning, not waiting, doing, none, want_to_play, not task_is_hard]	

The effect has been reached, it is a match!

At this point, the plan execution proceeds via `WarningCleanUp`, a scripted action that warns the user the time is about to elapse.

The following interactive action reported is: `PutAway`. Here the focus shifts to ensure the user is properly putting away the game and has acknowledged the warning received.

Log	Translation	Commentary
U: Va bene Pepper, metto via le carte	U: Alright Pepper, I'm putting the cards away.	Pepper successfully confirms that the user agrees to put away the game, as the predicate <code>agreed</code> is added.
R: Giulia, mi fa piacere che abbiamo trovato un accordo!	R: Giulia, I'm glad we found an agreement!	
P: [not checking, not available, warned, agreed]	P: [not checking, not available, warned, agreed]	
U: Riposiziono le carte	U: I'm repositioning the cards.	The predicate <code>doing</code> is added as the user engages in the task of putting away the cards. The robot continues to monitor the user's actions.
R: Giulia, hai deciso di sistemare tutto perché hai finito?	R: Giulia, did you decide to tidy everything up because you have finished?	
P: [not checking, not available, warned, agreed, doing]	P: [not checking, not available, warned, agreed, doing]	

Log	Log	Commentary
U: Sì	U: Yes.	Here it is interesting to see
R: Giulia, mi fa piacere sapere che hai finito e messo via tutto con cura!	R: Giulia, I'm glad to hear that you finished and put everything away carefully!	that even though the minimum time has elapsed it is not a <i>match</i> until the predicate doing
P: [not checking, not available, warned, agreed, doing, none]	P: [not checking, not available, warned, agreed, doing, none]	is negated. This will be better discussed in Section 5.2, since the predicate should describe the act of playing more than the general execution of an action.
U: Sì, infatti non lo sto facendo più	U: Yes, in fact, I'm not doing it anymore.	
R: Giulia, sono felice che siamo sulla stessa lunghezza d'onda!	R: Giulia, I'm happy that we are on the same wavelength!	
P: [not checking, not available, warned, agreed, not doing, none]	P: [not checking, not available, warned, agreed, not doing, none]	

The effect has been reached, it is a match!

This verbal interaction concludes the logs of the sections since the following actions to be implemented do not require speech generation and predicate retrieval.

Strategy implemented

Once the speech generation, the predicates retrieval, and the matching mechanism have been validated, it is possible to specifically test the replanning system. The beginning of the interaction is the same as the one previously described. The initial plan is generated starting from the first domain uploaded in the robot.

```
[(interact game kid memory),
 (presenttask game kid memory),
 (wait game kid memory),
 (warningcleanup game kid memory),
 (putaway game kid memory),
 (confirmallgood game kid memory x1 x1),
 (goodjob game kid memory x1 x1)]
```

It is important to note that the first action, *Interact*, did not produce any different results from those previously reported. Consequently, this log analysis begins with the third action of the plan, *Wait*, where the first signs of the necessity of replan are marked.

Log	Translation	Commentary
U: Pepper possiamo parlare mentre gioco?	U: Pepper, can we talk while I play?	The robot detects a shift in user engagement, recognizing
R: Giulia, giochiamo insieme!	R: Giulia, let's play together!	that the user is no longer interested in continuing the game.
P: [warning, not waiting, want_to_play]	P: [warning, not waiting, want_to_play]	This leads to a <i>go</i> (see Section 3.1) of the current predicates, with <i>not doing</i> acting as the trigger. This match with the preconditions of an alternative action signals the need for replanning.
U: No tranquillo, non voglio più giocare	U: No, don't worry, I don't want to play anymore.	
R: Giulia, perché sei così contraria?	R: Giulia, why are you so opposed?	
P: [warning, not waiting, want_to_play, not agreed, not doing]	P: [warning, not waiting, want_to_play, not agreed, not doing]	

Replan is needed

In response to this interaction, a new plan is formulated, which includes the following action: (*discovertom1 game kid memory*)

Log	Translation	Commentary
U: Perché questo gioco è troppo difficile!	U: Because this game is too difficult!	The robot investigates the user's mental state. Based on the user's answer to the previous question the robot updates the predicates adding <i>task_is_hard</i> which newly triggers the replan.
R: Giulia, cosa ti rende difficile questo gioco?	R: Giulia, what makes this game difficult?	
P: [investigating, task_is_hard]	P: [investigating, task_is_hard]	

Replan is needed

The predicates added match the preconditions of the *ConfirmHardTask* action, indicating the behavioral function *Escape* necessitating the specific strategy *StrategyHardTask*. The newly retrieved plan looks as follows:

```
[(confirmhardtask game kid memory x1 x1),
(strategyhardtask game kid memory x1 x1),
(wait game kid memory),
(warningcleanup game kid memory),
(putaway game kid memory),
(confirmallgood game kid memory x1 x1),
(goodjob game kid memory x1 x1)]
```

As mentioned in Section 3.5, the `ConfirmHardTask` action does not need to be physically executed by the robot, as its primary function is to validate the user ToM newly acquired by the robot. Then the strategy intervention occurs within `StrategyHardTask` action, where Pepper initiates a revised approach by stating: “*Ho capito. Forse è un po’ troppo difficile.*” This acknowledgment allows the interaction to transition smoothly into the new strategy phase, addressing the perceived difficulty and adapting the course of action accordingly.

Then the interaction proceeds back to the `Wait` action, while the user plays and the robot checks them. It is important to highlight that the game the user is now engaging with is a simplified version of the previous one, where all the cards are turned face-up.

Log	Translation	Commentary
U: Ma così è troppo facile! Non lo faccio più!	U: But that makes it too easy! I’m not doing it anymore!	The user clearly expresses that the game has become too easy,
R: Giulia, come mai hai deciso di non farlo?	R: Giulia, why did you decide not to do it?	not wanting to continue playing. A replan is necessary to
P: [warning, not waiting, not doing]	P: [warning, not waiting, not doing]	investigate the user’s mental state more deeply.

Replan is needed

As a result, the plan is once again updated to: [(discovertom1 game kid memory)]

Log	Translation	Commentary
U: Perché è troppo noioso, è proprio facilissimo!	U: Because it’s too boring; it’s just too easy!	Based on the user’s latest response the robot understands
R: Giulia, cosa ti rende così noioso questo compito?	R: Giulia, what makes this task so boring for you?	that the task has become too boring for the user and adds
P: [investigating, task_is_boring]	P: [investigating, task_is_boring]	the predicate <code>task_is_boring</code> , triggering another replan.

Replan is needed

Similarly to the previous replan the predicates were added to match the preconditions of the `ConfirmBoringdTask` action, newly indicating the behavioral function *Escape* necessitating the specific strategy `StrategyBoringTask`. As a result, the plan is updated as follows:

```
[(confirmboringtask game kid memory x1 x1),
(strategyboringtask game kid memory x1 x1),
(wait game kid memory),
(warningcleanup game kid memory),
(putaway game kid memory),
(confirmallgood game kid memory x1 x1),
(goodjob game kid memory x1 x1)]
```

After the ToM of the user is acknowledged by the `ConfirmBoringdTask`, and the game is made more interesting by the application of the strategy, the follow-up action is `Wait`. This time though, the interaction proceeds smoothly until the action conclusion. Then the subsequent behavior executed is `WarningCleanUp` ending by warning the user the time to play is ended and warning them it is time to `PutAway`.

Log	Translation	Commentary
U: Non sono d'accordo! Voglio continuare a giocare. Anzi lo faccio.	U: I disagree! I want to keep playing. I'm going to do it.	The robot is aware the user does not agree with the request of putting away the game, therefore the predicates are updated by adding <code>not agreed</code> , <code>want_to_play</code> and <code>doing</code> , triggering a new replan.
R: Giulia, sei sicura di non voler mettere via tutto?	R: Giulia, are you sure not to put everything away?	
P: [not checking, not available, warned, not agreed, want_to_play, doing]	P: [not checking, not available, warned, not agreed, want_to_play, doing]	

Replan is needed

The updated plan now consists of `[(discovertom2 game kid memory)]` which has the same function of `discovertom1`, but is triggered but with different preconditions.

Log	Translation	Commentary
U: Ti ho detto che voglio giocare ancora!	U: I told you I want to play more!	The robot confirms that the user wants to continue playing and updates the predicates accordingly.
R: Giulia, cosa ti piace di più del gioco che stiamo facendo?	R: Giulia, what do you like most about the game we're playing?	
P: [investigating, warned, want_to_play]	P: [investigating, warned, want_to_play]	

Replan is needed

The added predicate `want_to_play` match the preconditions of the `ConfirmWant2Play1` action, indicating the behavioral function *Gain a tangible* which then triggers the adapt strategy `StrategyWant2Play`, this is represented by the retrieved plan as follows:

```
[(confirmwant2play1 game kid memory x1 x1),
(strategywant2play game kid memory x1 x1),
(goodjobstrategy game)]
```

The experimental session then ends with the scripted behavior `GoodJobStrategy`, which greets the user.

4.3 Data Representation

The logs mentioned in the previous section also include information about the *Replan Time*, which is presented in Table 4.1. This Table reports the time required to generate a new plan from the same problem for both the client located on the robot (in the left column) and the client on the same machine as the server (right column).

Plan Retrieval Time Robot (s)	Plan Retrieval Time Local (s)
0.138749	0.122235
0.080601	0.110175
0.092073	0.111422
0.100321	0.114035
0.089148	0.085811
0.090264	0.080705
0.092446	0.084049

Table 4.1 Plan retrieval time (s).

The *Replan time* is measured from the moment the need for replanning is detected until the new plan is received by the client. To compare the use of cloud resources with a local implementation of planning capabilities, the same PDDL problem, automatically generated whenever replanning is needed, is sent both to a Planner Engine running on the server and to a Planning Engine executed locally on the client. The planning times range from 0.09 seconds to 0.13 seconds when using the Server-based implementation, and from 0.08 seconds to 0.12 seconds in the local implementation. These results show that using cloud resources does not reduce the time required for planning, and this may be due to the relative simplicity of the plans generated in this scenario. Rather, the average time when using the cloud can even be longer because of Client-to-Server communication, even though this does not cause visible delays during the interaction with the robot.

Regarding the *Cloud Response Time*, the graph in Figure 4.3 illustrates the correlation between the time required to process a user's sentence and the number of characters in the sentence. The Figure shows both the time required to retrieve a response sentence used to reply to the user (blue) and the time taken to retrieve grounded predicates describing the child's activity and mental state (red).

The Figure highlights a difference in the number of characters processed between the two tasks. In our current implementation, the instructions given to OpenAI for generating a reply are more complex, as they need to specify additional parameters related to the current context and the characteristics of the person the robot is interacting with. The differing character count in the prompts is reflected in the average cloud response time, which is approximately 1 second for sentence retrieval and about 0.75 seconds for predicate retrieval. Both values are significantly lower than the 2 seconds considered acceptable for human conversation (49) (50), even if longer delays (≈ 3 seconds) are sometimes experienced.

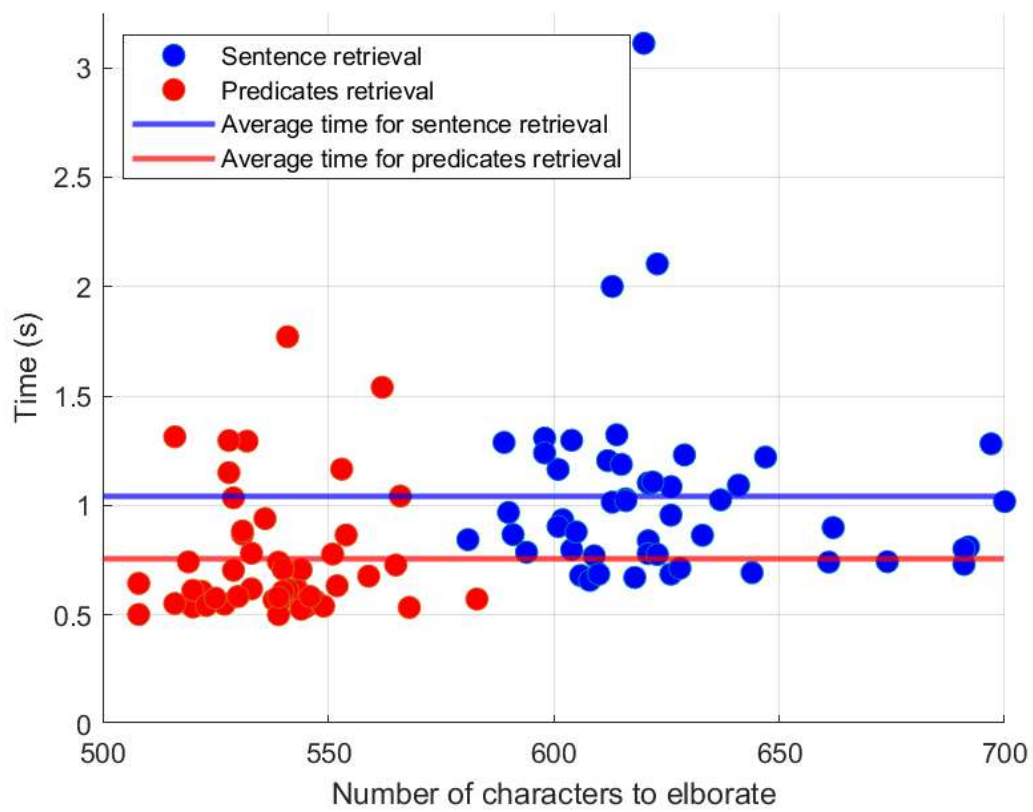


Figure 4.3 Cloud response time vs. number of characters in the user's sentence.

Chapter 5

Conclusion

5.1 Summary of Findings

As highlighted in the introduction, current research underscores the importance of designing sociable robots that can integrate seamlessly into daily life and sustain user engagement over extended periods. This study presents a framework for enhancing child-robot interaction, grounded in the principles of ABA, Applied Behavior Analysis. The system architecture introduces an innovative framework capable of real-time replanning to dynamically adapt to user behavior.

This project aims to address this gap by developing a flexible architecture that responds fluidly to users' evolving needs and circumstances. This has been made possible through the use of tools such as the PDDL framework, REST APIs, and OpenAI models. These tools not only facilitate smooth communication but also enable real-time adaptation of the framework, which is a key strength of this research. By bridging psychological theory with practical application, the project demonstrates the potential of social robots to effectively influence and enhance human behavior.

Nevertheless, it has been successfully demonstrated that the proposed framework operates as intended, delivering encouraging results in real-time adaptation and interaction. In particular, an education specialist with expertise in classroom settings reviewed the logs and confirmed that the participant's behavior and the robot's corresponding assessments were contextually appropriate, providing a positive evaluation of the *ABA implementation*. In addition, the metrics for *Replan Time* and *Cloud Response Time* were obtained with success, guaranteeing a response time of less than two seconds. This quick response is essential for preserving smooth interactions between the user and the robot since it reduces latency and improves the user experience in general. The robot's capacity to quickly reorganize and

react enables it to efficiently adjust to the changing dynamics of child-robot interactions, maintaining its engaging and contextually relevant character throughout the learning process.

5.2 Future Directions

Looking ahead, several key directions for future research and development emerge. The first step involves conducting experiments with recruited children. In collaboration with Allegra Bixio, an educational expert, the experimental protocol has already been outlined and can be found in Appendix B.

Moreover, from the underlying theoretical point of view, several potential limitations need consideration. ABA-based interventions are often criticized for focusing on behavior modification rather than supporting social acceptance of target behaviors in individuals with neurodevelopmental disorders (51). For this reason, the model's primary goal is to support behavior management in educational settings, rather than assisting a long-term therapeutic process.

Furthermore, this study does not account for setting events in antecedent assessment (e.g., internal and external conditions of the child, like physical discomfort or changes in the environment)(52). It does not evaluate children's acceptance of the robot, motivation, or attention during the interaction. These factors are crucial for conducting a more precise functional analysis and for supporting children's well-being while interacting with the robot.

On the other hand, the practical implementation could benefit from several enhancements. First, optimizing the system's overall efficiency and user experience requires improving the mechanism for checking predicates. This could involve relocating the predicate-checking function to a different method to accommodate various predicate retrievers, such as robot sensors and vision recognition modules.

Additionally, to enhance user engagement, the robot could initiate conversations instead of waiting for user prompts, and there could be significant direct body interactions.

Moreover, further development of the planner could enhance the project's practical applicability. For instance, conducting an extensive study on predicate names would facilitate smoother interactions, while a broader range of actions would cover more scenarios effectively. Additionally, an improved study on prompt design could lead to smoother interactions and help avoid undesirable scenarios.

These enhancements will not only improve user experience but also broaden the system's functionality, making it more versatile and effective in real-world applications.

References

- [1] Detlef Urhahne and Lisette Wijnia. Theories of Motivation in Education: an Integrative Framework. *Educational Psychology Review*, 35(2):45, June 2023.
- [2] Brian Scassellati. Theory of Mind for a Humanoid Robot.
- [3] Thomas Pierrot, Guillaume Ligner, Scott Reed, Olivier Sigaud, Nicolas Perrin, Alexandre Laterre, David Kas, Karim Beguir, and Nando de Freitas. Learning Compositional Neural Programs with Recursive Tree Search and Planning, 2019. Version Number: 2.
- [4] M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, July 2006. arXiv:1109.6051 [cs].
- [5] Cristine L. Chandler and James P. Connell. Children’s intrinsic, extrinsic and internalized motivation: A developmental study of children’s reasons for liked and disliked behaviours. *British Journal of Developmental Psychology*, 5(4):357–365, November 1987.
- [6] Omar Mubin, Catherine J. Stevens, Suleman Shahid, Abdullah Al Mahmud, and Jian-Jie Dong. A REVIEW OF THE APPLICABILITY OF ROBOTS IN EDUCATION. *Technology for Education and Learning*, 1(1), 2013.
- [7] Justin B Leaf, Joseph H Cihon, Ronald Leaf, John McEachin, Nicholas Liu, Noah Russell, Lorri Unumb, Sydney Shapiro, and Dara Khosrowshahi. Concerns about aba-based intervention: An evaluation and recommendations. *Journal of autism and developmental disorders*, 52(6):2838–2853, 2022.
- [8] Barry S Parsonson. Evidence-based classroom behaviour management strategies. *Kairaranga*, 13(1):16–23, 2012.
- [9] Ramy Hammady and Sylvester Arnab. Serious Gaming for Behaviour Change: A Systematic Review. 2022.
- [10] J. Broekens, M. Heerink, and H. Rosendal. Assistive social robots in elderly care: a review. *Gerontechnology*, 8(2):94–103, April 2009.
- [11] Amol Deshmukh, Sooraj K Babu, Unnikrishnan R, Shanker Ramesh, Parameswari Anitha, and Rao R. Bhavani. Influencing Hand-washing Behaviour With a Social Robot: HRI Study With School Children in Rural India. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1–6, New Delhi, India, October 2019. IEEE.

- [12] Richard M. Ryan and Edward L. Deci. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1):54–67, January 2000.
- [13] Joana Galvão Gomes Da Silva, David J Kavanagh, Tony Belpaeme, Lloyd Taylor, Konna Beeson, and Jackie Andrade. Experiences of a Motivational Interview Delivered by a Robot: Qualitative Study. *Journal of Medical Internet Research*, 20(5):e1116, May 2018.
- [14] Steven E. Stern, John W. Mullennix, Corrie-lynn Dyson, and Stephen J. Wilson. The Persuasiveness of Synthetic Speech versus Human Speech. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 41(4):588–595, December 1999.
- [15] Wenxuan Mou, Martina Ruocco, Debora Zanatto, and Angelo Cangelosi. When Would You Trust a Robot? A Study on Trust and Theory of Mind in Human-Robot Interactions. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 956–962, Naples, Italy, August 2020. IEEE.
- [16] Giunchiglia E. Recchiuto C. Sgorbissa A. D’Angelo I., Morocutti L. Nice and Nasty Theory of Mind for Social and Antisocial Robots. *32nd IEEE International Conference on Robot and Human Interactive Communication*, 2023.
- [17] Antonella Marchetti, Cinzia Di Dio, Angelo Cangelosi, Federico Manzi, and Davide Massaro. Developing ChatGPT’s Theory of Mind. *Frontiers in Robotics and AI*, 10:1189525, May 2023.
- [18] Marta Romeo, Peter E. McKenna, David A. Robb, Gnanathusharan Rajendran, Birthe Nessel, Angelo Cangelosi, and Helen Hastie. Exploring Theory of Mind for Human-Robot Collaboration. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 461–468, Napoli, Italy, August 2022. IEEE.
- [19] Susan Michie and Robert West. Behaviour change theory and evidence: a presentation to Government. *Health Psychology Review*, 7(1):1–22, March 2013.
- [20] Susan Michie, Michelle Richardson, Marie Johnston, Charles Abraham, Jill Francis, Wendy Hardeman, Martin P. Eccles, James Cane, and Caroline E. Wood. The behavior change technique taxonomy (v1) of 93 hierarchically clustered techniques: building an international consensus for the reporting of behavior change interventions. *Annals of Behavioral Medicine*, 46(1):81–95, 2013. Research Support, Non-U.S. Gov’t.
- [21] Fabrizio Mulas, Salvatore Carta, Paolo Pilloni, and Matteo Manca. Everywhere run: a virtual personal trainer for supporting people in their running activity. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, pages 1–2, Lisbon Portugal, November 2011. ACM.
- [22] Rodrigo De Oliveira and Nuria Oliver. TripleBeat: enhancing exercise performance with persuasion. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 255–264, Amsterdam The Netherlands, September 2008. ACM.

- [23] Falk Lieder, Mike Prentice, and Emily R. Corwin-Renner. An interdisciplinary synthesis of research on understanding and promoting well-doing. *Social and Personality Psychology Compass*, 16(9):e12704, September 2022.
- [24] Kohei Ogawa, Christoph Bartneck, Daisuke Sakamoto, Takayuki Kanda, Tetsuo Ono, and Hiroshi Ishiguro. Can an android persuade you? In *RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication*, pages 516–521, Toyama, Japan, September 2009. IEEE.
- [25] Ilaria Baroni, Marco Nalin, Mattia Coti Zelati, Elettra Oleari, and Alberto Sanna. Designing motivational robot: How robots might motivate children to eat fruits and vegetables. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 796–801, Edinburgh, UK, August 2014. IEEE.
- [26] Giovanna Castellano, Berardina De Carolis, Francesca D’Errico, Nicola Macchiarulo, and Veronica Rossano. PeppeRecycle: Improving Children’s Attitude Toward Recycling by Playing with a Social Robot. *International Journal of Social Robotics*, 13(1):97–111, February 2021.
- [27] Vijay Chidambaram, Yueh-Hsuan Chiang, and Bilge Mutlu. Designing persuasive robots: how robots might persuade people using vocal and nonverbal cues. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 293–300, Boston Massachusetts USA, March 2012. ACM.
- [28] Jaap Ham. Combining Robotic Persuasive Strategies: The Persuasive Power of a Storytelling Robot that Uses Gazing and Gestures. 2015.
- [29] Donato Malerba, Annalisa Appice, Paolo Buono, Giovanna Castellano, Berardina De Carolis, Marco De Gemmis, Marco Polignano, Veronica Rossano, and Lynn Margaret Rudd. Advanced Programming of Intelligent Social Robots. *Journal of e-Learning and Knowledge Society*, page Vol 15 No 2 (2019): Journal of eLearning and Knowledge Society, June 2019. Publisher: [object Object].
- [30] Aditi Ramachandran, Chien-Ming Huang, Edward Gartland, and Brian Scassellati. Thinking Aloud with a Tutoring Robot to Enhance Learning. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 59–68, Chicago IL USA, February 2018. ACM.
- [31] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *SCIENCE ROBOTICS*, 2018.
- [32] Daniel Leyzberg, Samuel Spaulding, and Brian Scassellati. Personalizing Robot Tutors to Individuals’ Learning Differences.
- [33] Martin Saerbeck, Tom Schut, Christoph Bartneck, and Maddy D. Janse. Expressive robots in education: varying the degree of social supportive behavior of a robotic tutor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1613–1622, Atlanta Georgia USA, April 2010. ACM.
- [34] Iolanda Leite, Ginevra Castellano, André Pereira, Carlos Martinho, and Ana Paiva. Empathic Robots for Long-term Interaction. 2014.

- [35] B R Schadenberg. Personalising game difficulty to keep children motivated to play with a social robot: A Bayesian approach. *Cognitive Systems Research*, 2017.
- [36] John O Cooper, Timothy E Heron, William L Heward, et al. *Applied behavior analysis*. Pearson/Merrill-Prentice Hall Upper Saddle River, NJ, 2007.
- [37] Yue Geo. A study on an applied behavior analysis-based robot-mediated listening comprehension intervention for ASD.
- [38] Ennio Cipani, Keven M Schock, et al. *Functional behavioral assessment, diagnosis, and treatment: A complete system for education and mental health settings*. Springer Publishing Company, 2010.
- [39] Kaneen B Geiger and Linda A LeBlanc. Function-Based Treatments for Escape-Maintained Problem Behavior: A Treatment-Selection Model for Practicing Behavior Analysts.
- [40] Paul Alberto and Anne C Troutman. *Applied behavior analysis for teachers*. Pearson Upper Saddle River, NJ, 2013.
- [41] Eva Yin-han Chung. Robot-Mediated Social Skill Intervention Programme for Children with Autism Spectrum Disorder: An ABA Time-Series Study. *International Journal of Social Robotics*, 2021.
- [42] Dr Barry S Parsonson. Evidence-based Classroom Behaviour Management Strategies. 13(1), 2012.
- [43] V A Ziparo, L Iocchi, P Lima, D Nardi, and P F Palamara. A Framework for Collaboration and Coordination in Multi-Robot Systems.
- [44] Luca Iocchi, Laurent Jeanpierre, Maria T Lazaro, and Abdel-Ilah Mouaddib. A Practical Framework for Robust Decision-Theoretic Planning and Execution for Service Robots.
- [45] E Sebastiani, R Lallement, R Alami, and L Iocchi. Dealing with On-Line Human-Robot Negotiations in Hierarchical Agent-Based Task Planner.
- [46] Vladimir Estivill-Castro and Jonathan Ferrer-Mestres. Path-finding in dynamic environments with PDDL-planners.
- [47] Sarah Dettmer, Richard L Simpson, Brenda Smith Myles, and Jennifer B Ganz. The use of visual supports to facilitate transitions of students with autism. *Focus on autism and other developmental disabilities*, 15(3):163–169, 2000.
- [48] Lucrezia Grassi, Carmine Tommaso Recchiuto, and Antonio Sgorbissa. Enhancing LLM-Based Human-Robot Interaction with Nuances for Diversity Awareness.
- [49] Zhenhui Peng, Kaixiang Mo, Xiaogang Zhu, Junlin Chen, Zhijun Chen, Qian Xu, and Xiaojuan Ma. Understanding user perceptions of robot’s delay, voice quality-speed trade-off and gui during conversation. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–8, 2020.

-
- [50] Toshiyuki Shiwa, Takayuki Kanda, Michita Imai, Hiroshi Ishiguro, and Norihiro Hagita. How quickly should communication robots respond? In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 153–160, 2008.
- [51] Justin B Leaf. Concerns About ABA-Based Intervention: An Evaluation and Recommendations. *Journal of Autism and Developmental Disorders*, 2022.
- [52] Rose Iovannone, Cynthia Anderson, and Terrance Scott. Understanding setting events: What they are and how to identify them. *Beyond Behavior*, 26(3):105–112, 2017.
- [53] Bryan J Weiner, Cara C Lewis, Cameo Stanick, Byron J Powell, Caitlin N Dorsey, Alecia S Clary, Marcella H Boynton, and Heather Halko. Psychometric assessment of three newly developed implementation outcome measures. *Implementation science*, 12:1–12, 2017.

Appendix A

Detailed Description of PDDL Domain Elements

A.1 Predicates

A.1.1 Agent State Predicates

- (doing ?a - agent ?t - task):
Indicates that agent ?a is currently working on task ?t.
- (agreed ?a - agent):
Denotes that agent ?a has agreed to what is asked.
- (task_is_hard ?a - agent ?t - task):
Specifies that task ?t is considered hard by agent ?a.
- (task_is_boring ?a - agent ?t - task):
Specifies that task ?t is considered boring by agent ?a.
- (want_to_play ?a - agent ?t - task):
Indicates that agent ?a wants to engage with task ?t.
- (touching ?a - agent ?o - obj):
Indicates that agent ?a is in contact with object ?o.

A.1.2 Robot State Predicates

- (waiting):
Represents that the robot is in a waiting state.
- (checking):
Represents that the robot is in a checking state.
- (investigating):
Represents that the robot is investigating the user ToM.
- (warning):
Represents that the robot is issuing a warning.
- (warned):
Indicates that the robot has issued a warning.

A.1.3 Task State Predicates

- (ready ?t - task):
Indicates that task ?t is ready to start.
- (start ?t - task):
Represents that task ?t has started.
- (finish ?t - task):
Indicates that task ?t is completed.
- (strategy_applied ?t - task):
Represents that a strategy has been applied to complete the task ?t.

A.1.4 Tangible State Predicates

- (available ?o - obj):
Indicates that object ?o is available.

A.1.5 Theory of Mind Predicates

- (belief ?i2 - id ?a - agent ?i1 - id):
Represents that agent ?a holds the belief ?i1. This is identified by ?i2.

- `(is_all_good ?i - id ?t - task)`:
Indicates that task ?t is considered to be all good. This is identified by ?i.
- `(is_hard ?i - id ?t - task)`:
Represents that task ?t is considered hard. This is identified by ?i.
- `(is_boring ?i - id ?t - task)`:
Indicates that task ?t is considered boring. This is identified by ?i.
- `(will_to_continue ?i - id ?t - task)`:
Represents that there is the will to continue with task ?t. This is identified by ?i.
- `(is_true ?i - id)`:
Represents that identifier ?i is true.

A.2 Actions

A.2.1 Interact

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (ready ?t) && (not (available ?o))
- **Effect:** (not (ready ?t)) && (start ?t)
- **Description:** This action allows the robot to verbally interact with an agent.

A.2.2 PresentTask

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (start ?t) && (not (available ?o))
- **Effect:** (not (start ?t)) && (available ?o) && (doing ?a ?t) && (waiting)
- **Description:** This action makes the robot present a task to an agent.

A.2.3 Wait

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (waiting) && (available ?o) && (doing ?a ?t)
- **Effect:** (warning) && (not (waiting)) && (doing ?a ?t)
- **Description:** This action makes the robot check on the user ensuring they is doing the task.

A.2.4 WarningCleanUp

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (warning) && (available ?o) && (doing ?a ?t)
- **Effect:** (warned) && (not (warning)) && (checking) && (doing ?a ?t)
- **Description:** This action warns the user and counts down the time needed to end the task.

A.2.5 PutAway

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (warned) && (checking) && (doing ?a ?t) && (available ?o)
- **Effect:** (warned) && (not (checking)) && (not (doing ?a ?t)) && (not (available ?o)) && (agreed ?a)
- **Description:** This action completes the task by making the robot ensure the user has put away the object.

A.2.6 ConfirmAllGood

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (warned) && (agreed ?a) && (not (available ?o)) && (not (doing ?a ?t)) && (not (checking))
- **Effect:** (is_all_good ?i1 ?t) && (belief ?i1 ?a ?i2) && (is_true ?i2)
- **Description:** This action confirms that everything is good and updates beliefs accordingly.

A.2.7 GoodJob

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (is_all_good ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2)
- **Effect:** (finish ?t)
- **Description:** This action represents the successful completion of a task after confirming that it is all good.

A.2.8 GoodJobStrategy

- **Parameters:** ?t - task
- **Precondition:** (strategy_applied ?t)

- **Effect:** (finish ?t)
- **Description:** This action finishes the task if a strategy has already been applied to it.

A.2.9 DiscoverTOM1

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (not (doing ?a ?t)) && (warning)
- **Effect:** (investigating) && (finish ?t)
- **Description:** This action initiates an investigation into a task when an agent is not currently doing it.

A.2.10 DiscoverTOM2

- **Parameters:** ?t - task, ?a - agent, ?o - obj
- **Precondition:** (doing ?a ?t) && (not (agreed ?a)) && (not (checking)) && (not (available ?o)) && (warned)
- **Effect:** (investigating) && (finish ?t) && (warned)
- **Description:** This action initiates an investigation into a task when an agent is not currently doing it.

A.2.11 ConfirmHardTask

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (investigating) && (task_is_hard ?a ?t)
- **Effect:** (is_hard ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2) && (not (investigating))
- **Description:** This action confirms that a task is hard and updates the belief about this fact.

A.2.12 StrategyHardTask

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (is_hard ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2)
- **Effect:** (available ?o) && (doing ?a ?t) && (waiting)
- **Description:** This action applies a strategy to handle a hard task.

A.2.13 ConfirmBoringTask

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (investigating) && (task_is_boring ?a ?t)
- **Effect:** (is_boring ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2) && (not (investigating))
- **Description:** This action confirms that a task is boring and updates the belief about this fact.

A.2.14 StrategyBoringTask

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (is_boring ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2)
- **Effect:** (available ?o) && (doing ?a ?t) && (waiting)
- **Description:** This action applies a strategy to handle a boring task.

A.2.15 ConfirmWant2Play1

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (investigating) && (warned) && (want_to_play ?a ?t)
- **Effect:** (will_to_continue ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2) && (not (investigating))
- **Description:** This action confirms that an agent wants to play with a task and updates beliefs about continuing the task.

A.2.16 ConfirmWant2Play2

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (investigating) && (touching ?a ?o)
- **Effect:** (will_to_continue ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2) && (not (investigating))
- **Description:** This action confirms the intention to continue with a task when an agent is touching the object.

A.2.17 StrategyWant2Play

- **Parameters:** ?t - task, ?a - agent, ?o - obj, ?i1 - id, ?i2 - id
- **Precondition:** (will_to_continue ?i1 ?t) && (belief ?i2 ?a ?i1) && (is_true ?i2)
- **Effect:** (strategy_applied ?t)
- **Description:** This action applies a strategy to a task when an agent is willing to continue.

Appendix B

Experimental Protocol

To assess the model's effectiveness in educational settings, an experimental design comprising 20 child-robot interaction sessions is proposed. Education experts will retrospectively evaluate these sessions to assess the robot's accuracy in detecting behavior functions and the appropriateness of employed strategies. The hypotheses posit that the functions identified by the model will align with experts' judgments, and the strategies employed will be approved by them.

B.1 Participants

The study involves a total of 30 participants, including 20 children aged 8-9 years randomly selected from Italian primary schools and 10 education experts, ideally primary school teachers. Each child engages in a one-on-one interaction with the robot and is accompanied by a caregiver who is asked to intervene during the study. Education experts assess child-robot interactions by reviewing their video recordings.

B.2 Hypotheses

Based on the evaluation of child-robot interactions conducted by education experts, the following hypotheses are proposed:

H1: There will be a significant agreement between the behavior functions assessed by experts and those detected by the robot during the experiment. A high level of agreement indicates the accuracy of the robot's capability to interpret the child's behavior.

H2: The strategies proposed by the robot during interactions will be positively evaluated by experts. A strong endorsement of the strategies employed would demonstrate increased validation of the robot's applicability in educational environments.

These hypotheses explore the alignment between education expert assessments and robot performance, contributing to understanding the applicability of robotics in education.

B.3 Experimental Procedure

During the experiment at the university laboratory, each child will independently interact with the humanoid robot Pepper, which operates based on the architecture detailed in Section 3.1. The robot autonomously plans the sequence of actions based on how the interaction evolves, following the outlined general rules.

Pepper starts by engaging the child and explaining the memory game, including the rules and timing. The child has 10 minutes to complete the game and is instructed to leave the room when the time is up. During the game, Pepper interacts minimally, asking only if the game is progressing well. If the robot detects that the task is too difficult, it allows the child to play with fewer cards. Conversely, if the game seems too boring, Pepper suggests playing with more cards to increase the challenge.

Two minutes before the time expires, Pepper provides a reminder. When the time is up, the caregiver is asked to call the child to leave. If the child complies, Pepper thanks them and says goodbye. If the child resists, Pepper recognizes the child's desire to continue playing and implements the following strategies: first, it grants an additional two minutes of play with a timer, and then it asks the child to help clean up the cards. Once the cards are put away, Pepper waves goodbye, and the participants leave the room. Each session, lasting approximately 15-20 minutes, is recorded.

We clarify that in the final phase of the experiment, caregiver intervention is intended to encourage the child to engage with the robot. We hypothesize that the presence of a familiar person may influence the child's noncompliance more than the robot's instructions. This allows for a more effective test of the system, as it increases the likelihood that the robot will need to implement strategies to manage the child's behavior. Moreover, Pepper interacts with minimal social cues to avoid influencing the child's task performance. Finally, the strategies implemented are carefully recommended by both the authors and the literature review. Specifically, one of the authors is an education specialist with expertise in classroom settings and reported the main strategies used in such contexts. Some of these, such as using a timer as a visual support, adjusting tasks based on students' entry skills, and providing

praise after the expected behavior is performed, are well-documented in ABA and behavior management studies (8)(47).

Approval from the Ethics Committee of the University of Genoa and parental consent will be obtained.

B.4 Measurements

For *H1*, is aimed to evaluate the agreement between Pepper's identification of behavioral functions and experts' judgments across 20 recorded sessions. Two experts review each video to assess whether the child exhibits the functions *Gain a Tangible* and *Escape*. Videos include clickable commands for each function and an "End Function" command to signal the cessation of function detection. Experts provide judgments at any appropriate point during the videos. After collecting the responses, we calculate the degree of agreement between the two experts, followed by the degree of agreement between each expert and the robot. The mean of these results is then determined.

This data analysis is conducted for every 30-second segment of the video using a Weighted Cohen's K test. Each segment is labeled as either *No Function*, *Function 1*, or *Function 2* based on the buttons clicked by experts. The degree of agreement is computed and averaged across all segments, taking into account different combinations of responses between the two evaluators. For example, simultaneous identification of two distinct functions (*Function 1* and *Function 2*) is weighted more than when one rater identifies *Function 1* and the other identifies *No Function*.

For *H2*, we aim to assess the effectiveness of Pepper's strategies through expert evaluation. In the recorded videos where Pepper implements strategies, experts indicate which strategies they would choose in the same scenario. Afterward, experts complete a survey evaluating the acceptability, appropriateness, and feasibility (53) of Pepper's strategies.

This approach evaluates the accuracy of Pepper's behavioral function identification and expert perceptions of its strategies.