Tool-Use Robot Manipulation Tasks for Cooperative and Explainable Operations in Safety-Critical Domains

by

Emily Jean Sheetz

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2025

Doctoral Committee:

Emeritus Professor Benjamin Kuipers, Chair
Professor Joyce Chai
Professor Odest Chadwicke Jenkins
Dr. Kimberly Hambuchen, NASA Johnson Space Center

The path from dreams to success does exist.
May you have the vision to find it,
the courage to get on to it, and
the perseverance to follow it.
Wishing you a great journey.

—Dr. Kalpana Chawla

Emily Jean Sheetz

esheetz@umich.edu

ORCID iD: 0000-0002-7317-3442

To all of us who doubt ourselves and
our ability to overcome challenges.

Be brave and keep going.
We can do it.

Acknowledgments

I am grateful to so many people for their support during my graduate studies. Thank you to my advisor, Prof. Benjamin Kuipers, for his encouragement and guidance in developing my abilities to think critically and contribute to the research community. Thank you to my committee members—Prof. Joyce Chai, Prof. Odest Chadwicke Jenkins, and Dr. Kimberly Hambuchen—for their feedback and insights on my dissertation.

I am proud to have been a member of both the Intelligent Robotics Lab and the Laboratory for Progress. Heartfelt thank you to Liz Olson and Jorge Vilchis for all of our research discussions and comradery. Thank you to the CRLT-Engin Engineering Teaching Consultants and to the Academic Success Program for enabling me to make a positive impact on campus. Immense thank you to department staff for the support that makes everything we do possible. I sincerely appreciate the funding that I received from the Computer Science and Engineering Department, the College of Engineering, and NASA.

I am so grateful to NASA Johnson Space Center and the Dexterous Robotics Team for all of the time, resources, support, mentorship, and encouragement. I sincerely appreciate my mentors—Misha Savchenko, Emma Zemler, Mark Paterson, and Shaun Azimi—for teaching me and inspiring me every day. You all made the completion of my degree possible.

I am very fortunate to have many supportive mentors. Thank you to Prof. Michael Sostarecz, Prof. James Logan Mayfield, Prof. Jonathan Sprinkle, Prof. Saad Biaz, Dr. Audra Baleisis, Dr. Stephen Hart, Dr. Ana Huamán Quispe, and Dr. Steven Jorgenesen. You all supported me and inspired me to continue my degree at the times I needed you the most.

Jim Kramer, thank you for being one of my first friends in Texas. I miss you so much.

Thank you so much to my family for the support throughout my whole life. Mom and Dad, you have given me and taught me so much. My grandparents—Tony, Rita, Mickey, and Vicky—thank you for always being so proud of me. My sweet pea and best dog ever, Princess. Jessica, my sister, thank you for your humor and boldness, and for being there for me always.

I am so grateful to have friends who support me in all areas of my life. Misha Savchenko, Amy Fritz, Lauren Nilsson, Mina Kian, Andrea Ventola, Teresa French, Emma Vanderpool, Ethan Hager, Emily and Andrew Danes, Ryne Krejci, Katie Harrold, Sanjay Singapuram, Tyler Sypherd, Nate Hamilton—for everything you have done for me, thank you.

And to my sweet and sassy cat, Boogie. You are my best buddy. I love you so much.

TABLE OF CONTENTS

List of Figures

FIGURE

List of Tables

TABLE

List of Appendices

Abstract

To be effective in assistive tasks, robots need to be capable of performing tasks programmed by non-expert users. Tool-use and assembly tasks are of particular interest as assistive tasks because they present many challenges such as reasoning over interactions between multiple objects and performing complex manipulation behaviors. Considering safety-critical domains further complicates robot reasoning by constraining these manipulation tasks and requiring that robots perform tool-use tasks subject to a wide range of safety considerations.

In this dissertation, we address the problem of reliable autonomous tool manipulation in safety-critical domains. Our goal is to advance planning and execution capabilities in tool-use object manipulation tasks through simple explainable models, enabling robots to engage in dialogue about safety on human-robot teams. We address the following challenges for safely executing tool-use tasks: (1) autonomously composing multi-objective behaviors (actions that satisfy multiple goals); (2) robustly modeling tool grasps and generalizing grasps to novel tools; and (3) reasoning over and engaging in dialogue about safety while performing tasks in different domains.

To perform multi-objective manipulation tasks, we explore reasoning over *composable causality* in furniture assembly tasks. We expect robots to autonomously compose behaviors to achieve given objectives without solely relying on qualitative observations from expert programmers. To formalize the *composable causality* of multi-objective actions, we propose a *causal control basis*. The *causal control basis* annotates the elements of a typical control basis (a set of controller behaviors) with causal information describing how a multi-objective action functions in an assembly task.

The robot uses the *causal control basis* to estimate the likelihood that different compositions of behaviors achieve the intended effect. The *causal control basis* effectively reduces reliance on expert knowledge engineering for performing complex actions, making the execution of these behaviors more explainable.

To further improve dexterous robot manipulation, we explore grasp reflex modeling through tactile servoing for robustly achieving tool grasps in manipulation tasks. We propose a *grasp reflex model*, a simple explainable model that detects meaningful adjustable states describing the robot's end-effector pose relative to the tool being grasped. Our trained *grasp reflex model* identifies statistically significant variables from the end-effector data, and when deployed on the robot, we find that our *grasp reflex model* achieves one-shot tactile servoing on 6 novel tool instances. Our proposed *grasp reflex model* is simple enough to be explainable and is reliable and generalizable enough to be trusted in tool manipulation tasks in safety-critical domains.

Towards tool-use and manipulation in safety-critical problem domains, we suggest that humans and robots must challenge each other's assumptions, minimize overtrust, and characterize risks. To make robots active, trustworthy collaborators, we propose the *human-robot red teaming* paradigm for *safety-aware reasoning*. We demonstrate that a *human-robot red team* can engage in dialogue about safety and improve the team's understanding of a problem domain. From these interactions, the robot learns to plan to complete tasks safely and mitigate risks during task execution. *Safety-aware reasoning* allows the robot to reason over and perform tool-use manipulation tasks alongside a human user under varying definitions of safety.

Taken together, our work emphasizes the importance of minimal expert knowledge engineering, interactions with non-expert users, and explainable methodology and models for robot manipulation capabilities. These factors justify the trust human users place in robot systems, and enable robots to reliably perform complex manipulation tasks on human-robot teams in safety-critical problem domains.

Chapter 1

Introduction

1.1 Robots in Space Exploration and Extreme Environments

As robots become more intelligent, capable, and present in different environments, they face increased opportunities for risk or damage. In extreme environments with high-stakes applications, robots must be capable of addressing a wide variety of challenges and operate safely in difficult conditions.

Safety in robotics applications can have varying definitions depending on the anticipated hazards. As seen in Figure 1.1, many factors contribute to the risk of a task, including the environment, the presence of humans, and the level of robot autonomy. Low-risk applications include highly constrained environments, robots operating without humans nearby, and/or low levels of robot autonomy. High-risk applications include dynamic and unconstrained environments, robots operating alongside humans, and/or high levels of robot autonomy. Figure 1.2a depicts robots facing different environmental risks. Robots operating in an unconstrained, dynamic environment such as a lunar habitat will face more hazards than robots operating in a highly controlled environment such as a factory assembly line. Figure 1.2b shows robots facing risks induced by humans in the environment. A Mars rover may face environmental risks, but because there are no humans on the Martian surface, the risk of human injury is negligible. In contrast, a household robot is expected to operate in human environments alongside the human inhabitants of

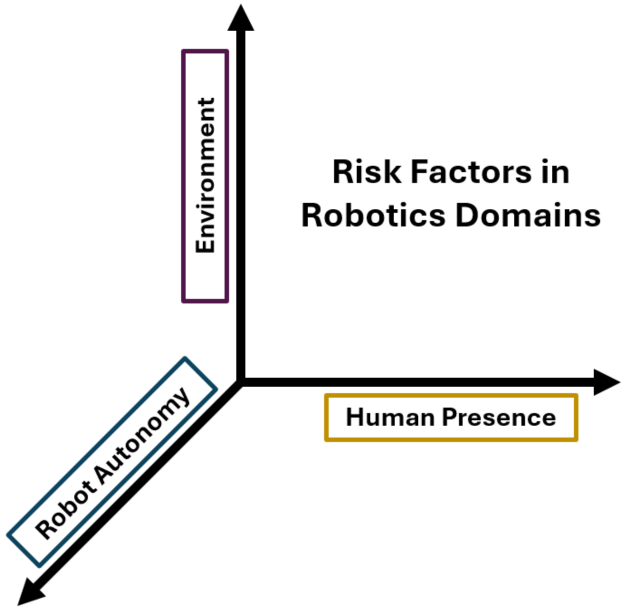


Figure 1.1: Several factors affect the risks in a problem domain, including environment, human presence, and level of robot autonomy. Figure 1.2 explores each risk factor in more detail.

the home, making the risk to human life significantly greater. Finally, Figure 1.2c portrays robots operating with different levels of autonomy. A robot expected to reason and act autonomously (even if that autonomy is supervised by or shared with human operators) will face more hazards than a robot that is teleoperated or fully controlled by a human operator. For example, Robonaut helped astronauts perform maintenance tasks on the International Space Station (ISS) but was always fully controlled and teleoperated by astronauts or ground control. Because the human operators had full control over the robot, Robonaut functioned like any other tool, meaning the ISS crew accepted the usual operating risks associated with human decision-making. In contrast, a robot operating with its own autonomy must be able to identify hazards, assess risks, and perform tasks safely the same way a human operator would.

(a) Low environmental risks include industrial assembly line robots [15] operating in highly constrained environments. High environmental risks include dynamic, unconstrained environments, such as robots operating in NASA's vision for consistent presence and extended habitation on the lunar surface [16].

(b) Robots operating without humans present face lower risks, such as the Mars rovers [17, 18, 19]. Human presence increases the risks, such as Valkyrie performing tool-use tasks [20] in terrestrial household environments.

(c) Teleoperation of Robonaut [21] onboard the International Space Station (ISS) lowers the level of robot autonomy and therefore lowers the operating risks. The shared autonomy of iMETRO [22, 23] increases the operating risks.

Figure 1.2: Factors that affect the risks in safety-critical robotics applications, as seen along several spectra. We consider hazards with respect to potential risk to human life, ranging from minor injury to potentially life threatening.

We consider robots performing assistive tasks in pursuit of space exploration to be towards the upper extreme of task domains in terms of safety requirements and risks that robots may face, as shown in Figure 1.2. Space exploration not only presents high-stakes hazards and risks to human life, but also presents unique robotics challenges [24] that must be faced. While in space, robots must perform manipulation tasks in micro-gravity [25]. Robots must work collaboratively on different types of teams, such as human-robot teams [26] and multi-robot teams. For example, intra-vehicular robots (IVRs) and extra-vehicular robots (EVRs) will cooperatively perform logistics and resupply tasks for pressurized rovers on the lunar surface [27, 28]. Working alongside humans [29] in space presents further challenges since robots are often commanded by flight control personnel or astronauts who likely do not have extensive programming or robotics experience. Furthermore, the great distances between ground control on Earth and astronauts and robots in space requires that robots be operated across great time delays [30, 31]. Space robotics explores unique challenges, which impact the robot's ability to perform tasks in a high-stakes environment while reasoning over safety to minimize risk to human life.

In order for robots to be trusted to operate in extreme safety-critical domains, they must face the unique challenges of high-stakes and high-risk tasks. Firstly, robots must have the ability to autonomously perform their commanded tasks. Manipulating objects and using tools for assembly or maintenance tasks requires advanced reasoning and manipulation capabilities. Secondly, robots must be able to operate safely in these challenging domains. Robots need to identify hazards and mitigate risks in order to perform tasks safely. Taken together, these robot capabilities must be flexible and adaptive to different robots operating under varied definitions of safety in extreme high-stakes domains.

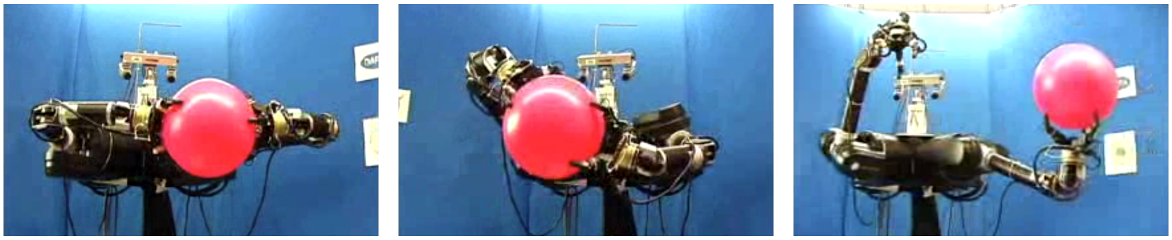
This dissertation addresses two key challenges to achieve safety reasoning in tool-use tasks: (1) learning planning policies and executing complex tool-use actions (Section 1.1.1), and (2) robot safety awareness and risk mitigation (Section 1.1.2).

1.1.1 Planning and Executing Actions for Tool-Use Tasks

Early robotics applications involved robots performing simple pick-and-place tasks in highly controlled environments such as factory assembly lines. However, real-world applications require robots to perform diverse types of interactions such as pushing, pulling, and rotating objects [32]. An "affordance" is a property of an object that indicates a possible way to interact with that object given the constraints of the agent's embodiment and the environment [33]. Object affordances can be thought of as "action possibilities" or "opportunities for action" that can be perceived in a situation. Based on the perceived affordances, an agent can plan how to achieve a task by considering what "opportunities for action" it should or should not take. The agent must also be able to execute the afforded action in order to fully realize the task goals. Robotics research has long explored how robots can perceive affordances, plan actions, and execute the afforded actions for different tasks.

Tool-use tasks are especially challenging [34] because they require advanced planning and action execution capabilities. In terms of planning, tool manipulation requires robots to reason over complex task goals and interactions between the tools and other objects manipulated by the tool [13, 35]. In terms of action execution, using tools requires the robot to ground object understanding in robot action [36], intentionally interact with the tools in order to achieve the larger manipulation task [37, 38, 39], and perform complex motions [5, 40, 7, 41, 42, 43, 44] to carry out the tool-use behavior.

Complex tool-use actions often take the form of *multi-objective actions* [45, 46, 47, 48, 49, 50], which achieve multiple sub-goals with a single action. Figure 1.3 (from Platt *et al.* [1]) shows several examples of multi-objective actions, where the robot performs various reaching tasks while maintaining a secure grasp on a ball. As



(a) Two-handed reach. (b) Transfer grasp. (c) One-handed reach.

Figure 1.3: Examples of multi-objective actions from Platt *et al.* [1], reprinted with permission of the authors. The robot performs a series of tasks while keeping the ball in-hand, demonstrating its ability to reason over multiple sub-goals concurrently. The robot first achieves a two-handed grasp and moves the ball around the workspace while maintaining the secure grasp (Figure 1.3a). Then the robot transfers from a two-handed grasp to a one-handed grasp without dropping the ball (Figure 1.3b). The robot again moves the ball around the workspace while maintaining a secure one-handed grasp (Figure 1.3c).

another example, screwing in a screw requires *positioning* the screwdriver tip relative to the head of the screw, *aligning* the screwdriver with the screw, and *rotating* along an axis in order to drive the screw. Such multi-objective actions are executed by composing multiple behaviors from a *control basis* [45, 46, 47, 48], which is a set of controller behaviors that can be executed or combined to result in more complex actions than the basis behaviors. The combinations of these multi-objective behaviors are generally predefined by expert programmers to yield the expected robot behavior. Providing robots with the reasoning capabilities to autonomously compose controllers in manipulation tasks [6] remains an open area of research.

Especially in safety-critical domains, robots need to be capable of performing their commanded tasks. Our work explores methods to advance robot planning and execution of multi-objective tool-use behaviors with minimal knowledge engineering from experts.

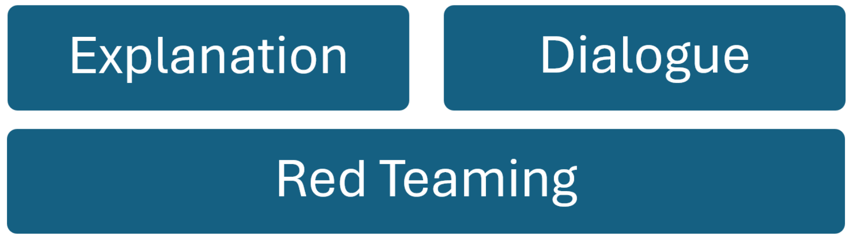
1.1.2 Safety Reasoning and Risk Mitigation

While improving robot capabilities is an immensely important research goal, many works do not consider how the robot should perform these capabilities safely [51].

This lack of safety reasoning is troubling, as there is clear consensus about the importance of safe operations, especially in environments where robots operate alongside humans [52, 53, 54, 55, 56, 57]. Researchers may not consider safety due to overtrust in the robot's capabilities and overtrust that the human operator will appropriately evaluate safety [58, 59].

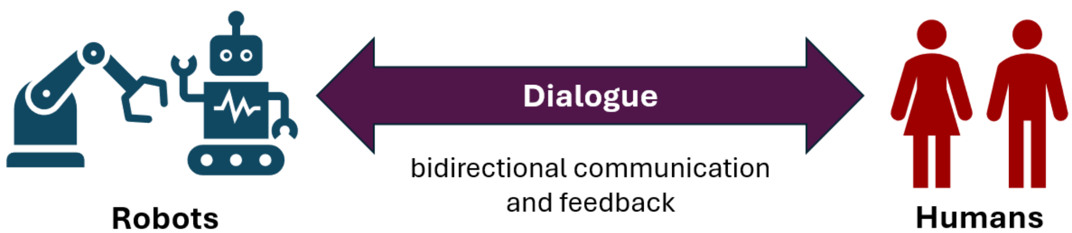
In collaborative safety-critical tasks such as space exploration, *trust* between humans and robots is crucial [60, 61, 62, 63]. Humans may mistrust robots for a variety of reasons. Unpredictable robot behaviors may not meet the humans' expectations. A lack of explanation on why the robot performed an action or made a particular assessment can lead to confusion for human operators [58]. Not only do such robot behaviors lead to mistrust and poorly calibrated trust, but these unpredictable, unreliable, and inexplicable robot behaviors also have the potential to be unsafe or create unsafe operating conditions [64]. In contrast, if a human better understands what the robot is doing and why, they are more likely to trust the robot appropriately.

To promote understanding within human-robot teams in safety-critical domains, we expect human and robot agents to communicate. In this dissertation, we explore the role of several key components for communication and shared understanding on human-robot teams. We refer to these collective components as the **C**ommunication and **U**nderstanding through **R**ed teaming, **E**xplanation, and **D**ialogue (CURED) Framework, as depicted in Figure 1.4. The three key aspects of the CURED Framework—explanation, dialogue, and red teaming—are seen in Figure 1.4a and described in the following sections—Section 1.1.2.1, Section 1.1.2.2, and Section 1.1.2.3, respectively. Section 1.1.2.4 describes the collective role of the CURED Framework components in our work.

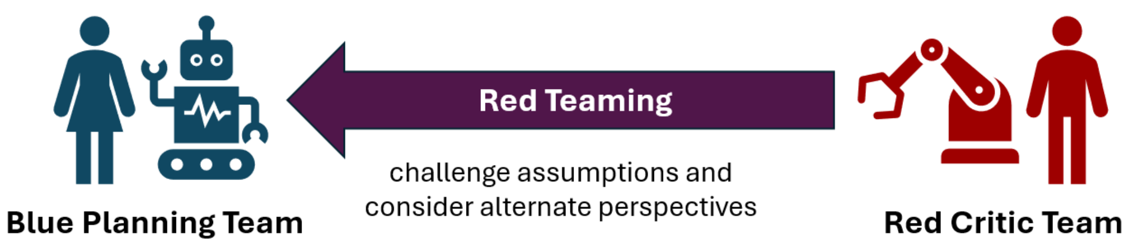


(a) The CURED Framework for communication achieves understanding on human-robot teams through three components: explanation, dialogue, and red teaming.

(b) **Explanation** allows the robot speaker(s) to achieve new understanding in the human listener(s) by describing shared task goals and plans of action.



(c) **Dialogue** enables the humans and robots to act as both speakers and listeners and engage in bidirectional communication through which they share feedback, questions, justifications, and clarifications.



(d) **Red teaming** is the process by which a red team challenges the blue team's assumptions about a situation, thereby encouraging the blue team to consider alternate perspectives. In our work, we propose *human-robot red teaming*, which allows both humans and robots to serve on the blue planning team or the red critic team at different times.

Figure 1.4: The **C**ommunication and **U**nderstanding through **R**ed teaming, **E**xplanation, and **D**ialogue (CURED) Framework.

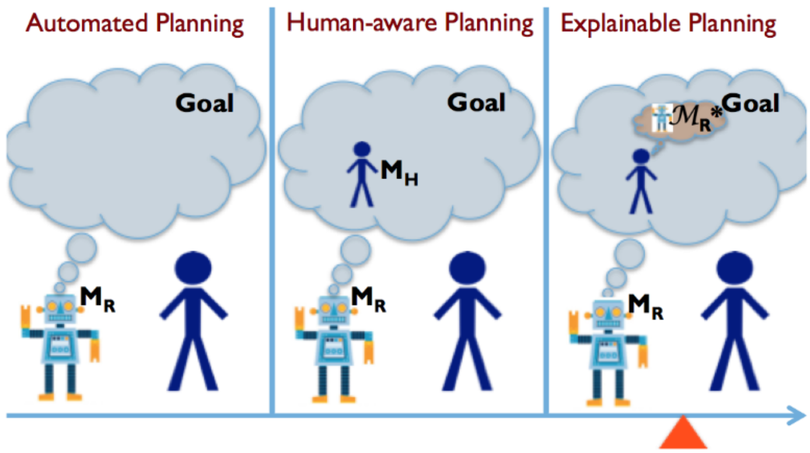


Figure 1.5: Explainable robot reasoning from Zhang *et al.* [2], reprinted with permission of the author. For explainable human-robot interactions, robots not only need their own mental models of the environment, action space, and task goals, but also need to understand the human's mental model. By considering how human mental models differ from its own, the robot can provide appropriate explanations for its reasoning [2].

1.1.2.1 Explanations

One important aspect of communication and promoting understanding on human-robot teams is for robots to *explain* their reasoning. Explanations, as depicted in Figure 1.4b, allow the speaker to achieve new understanding in the listener by describing shared task goals, plans of action, and justifications for how and why the plan is predicted to work. Many works explore explainable artificial intelligence and robotics [65, 66]. As seen in Figure 1.5 (from Zhang *et al.* [2]), in order for robots to adequately explain their reasoning over a task, they need to consider how their mental models differ from the mental models of human operators. While there can be a disconnect between explanations and the processes being explained [67], explanations can be a useful tool for often mistrusted robots to minimize the differences between human and robot mental models. Especially for non-expert users commanding robots in safety-critical domains—such as flight control personnel or astronauts using assistive robots in space exploration applications—explanations are immensely important for establishing shared understanding and eventually earning well calibrated trust. If our robots can earn the trust of their non-expert human users while reliably and safely performing assistive tool-use tasks, our methods are more likely to be flight certified and deployed, thereby allowing robots to further assist humanity in the most safety-critical applications.

One way to enable robot explanations is exploring methods that use *simple decision-making models*. Researchers theorize that human reasoning is probabilistic in nature [68]. Utilizing simple decision-making models [69, 70] can allow robot reasoning to approximately mimic human probabilistic reasoning. By using a model that minimizes the difference between human and robot mental models [2], we promote explainability of robot behaviors at all stages of operation [71, 72, 73, 74]. Explainable decision-making models are often simple algorithms that reduce expert knowledge engineering and promote understanding for non-expert users.

1.1.2.2 Dialogue

Another way to promote understandable robot operations is to have robots explain their decision-making through *dialogue*. Figure 1.4c demonstrates that dialogue enables bidirectional communication between speaker and listener, where both parties share explanations, feedback, questions, clarifications, and justifications. Engaging in dialogue to provide explanations [11] enables the robot to address differences between human and robot mental models through intuitive communication.

While we use natural language for communication and dialogue on human-robot teams, we are not contributing to research in understanding syntax or semantics for natural language processing. Instead, we limit our language processing capabilities to an “English-like” interface, and leave it to future work to further develop the natural language commands processed in safety-critical domains. Our work explores how these dialogue interactions can enable communication and improve understanding between conversation partners on a cooperative human-robot team. By promoting greater understanding between agents, explanations through dialogue can eventually lead to trust, empowering human-robot teams [26] to work together to achieve their task goals.

1.1.2.3 Red Teaming

It is important to carefully consider the hazardous conditions that may arise in safety-critical problem domains in order to assess and mitigate risks [75, 76]. *Red teaming* is a technique used to consider multiple perspectives, detect weaknesses, and reveal biases to inform decision making before disastrous outcomes occur [77, 78, 79, 80, 81, 82, 83, 84, 85], as seen in Figure 1.4d. Many domains use human red teams [86, 77, 78, 87, 83, 88], where humans work to identify hazardous conditions. More recently, *computational* red teams [85, 79, 81, 89, 90] have been used to automate the red teaming process and avoid human biases. Red teaming exercises are important for critically assessing a team's performance, challenging assumptions, considering alternate perspectives, and improving understanding.

1.1.2.4 The CURED Framework in Safety-Critical Domains

Taken together, the CURED Framework components enable improved communication and understanding on human-robot teams. In safety-critical problem domains, robots must anticipate and avoid disastrous outcomes through red teaming, explain their reasoning by using simple decision-making models, and engage in dialogue to share information and receive feedback. In this way, the components of the CURED Framework promote understanding and enable robots to earn the trust of human operators. We explore simple explainable algorithms for robot policies and propose *human-robot red teaming* (Figure 1.4d) as a strategy to avoid biases in risk assessment, allow robots to reason over safety, and engage in dialogue about risk mitigation.

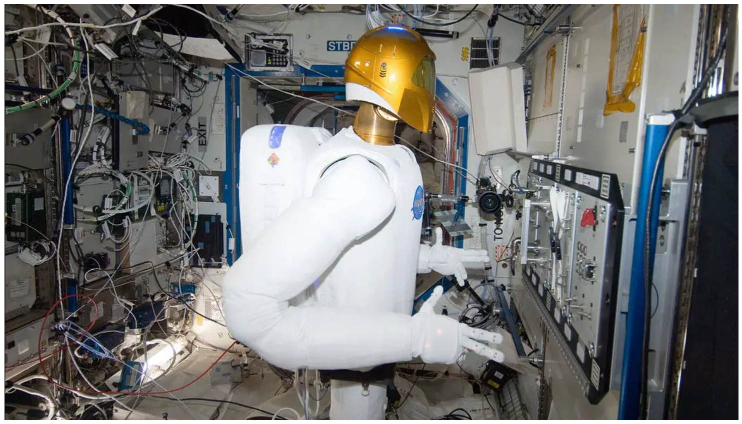
1.1.3 Thesis Statement

In this dissertation, our goal is to advance planning and execution capabilities in tool-use object manipulation tasks through simple explainable models, enabling robots to engage in dialogue about safety on human-robot teams operating in safety-critical domains. We need robots to perform assistive object manipulation, tool-use, and assembly tasks while reasoning over safety and mitigating risks as necessary to ensure safe operations.

We address key challenges of explainable safety reasoning in tool-use tasks, specifically: (1) planning over and executing complex object manipulation actions in tool-use tasks, (2) using simple explainable algorithms with minimal expert knowledge engineering to develop understanding on human-robot teams, and (3) engaging in dialogue about safety in order to mitigate risks in safety-critical domains. Methods for improving safety awareness, risk mitigation, and planning and executing complex object manipulation tasks must consider differing robots' capabilities and the unique challenges of varied safety-critical domains. To achieve this, we evaluate our methods primarily in two environments: terrestrial household environments and space exploration environments, as seen in Figure 1.6. These environments come with different risks, definitions of safety, and risk mitigation strategies. By considering how our proposed methodologies work on different robots in different environments, we explore how to generalize manipulation and safety reasoning capabilities to various high-stakes safety-critical problem domains.

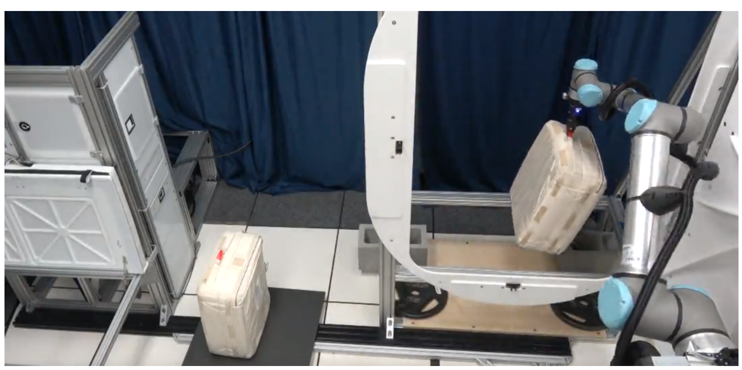
1.2 Statement of Dissertation Scope

To address the challenges of planning and executing tool-use affordances in safety-critical domains, we explore explainable methods that require minimal knowledge engineering and promote understanding on human-robot teams.



(a) Robonaut [91] performing routine tasks through teleoperation alongside astronauts on the International Space Station (ISS) [21].

(b) Valkyrie [92]—a humanoid robot operating in dynamic environments alongside human operators [29]—performing terrestrial tool-use and item retrieval task [20].



(c) iMETRO (Integrated Mobile Evaluation Testbed for Robotics Operations) [22, 93, 94] performing intra-vehicular cargo transfer task as would be performed in a lunar habitat [23, 95].

Figure 1.6: Whether in terrestrial tasks, in space applications [3] such as the International Space Station (ISS), or in dangerous domains such as lunar habitats [4], robots must be able to perform assistive tool-use and object manipulation tasks while reasoning over safety.

More specifically, this dissertation makes the following contributions:

1. **Composable Causality in Semantic Robot Programming** (Chapter 3) [96]: We propose a *causal control basis* to improve robot planning over complex multi-objective manipulation actions in assembly and tool-use tasks. The *causal control basis* annotates the elements of a typical control basis with causal information—such as the intended effects of a combined behavior—describing how the composed action should function within an assembly task. By reasoning over the intended effects, the robot can predict how to combine the controllers autonomously without solely relying on qualitative observations from expert programmers. We assume the robot can perceive the objects in the environment and their affordances. In furniture assembly tasks, we demonstrate that the *causal control basis* can be used to predict the probability of successful multi-objective part connections, and achieve a success rate of about 86% across 20 different assembly trials involving a total of 70 part connection attempts. The *causal control basis* effectively reduces reliance on expert knowledge engineering for performing complex actions, making the execution of these behaviors more explainable.

2. **Multi-Fingered End-Effector Grasp Reflex Modeling for One-Shot Tactile Servoing in Tool Manipulation Tasks** (Chapter 4) [97]: In order to loosen our perception assumptions, we explore *grasp reflex modeling* through tactile servoing for robustly achieving tool grasps for manipulation tasks. Inspired by the human grasp reflex as an example of touch-driven control, we propose a *grasp reflex model*, a simple explainable model that detects meaningful adjustable states describing the robot's end-effector pose relative to the tool being grasped. Our trained *grasp reflex model* identifies statistically significant variables from the end-effector data. In simulated grasping tasks, the *grasp reflex model* achieves about 89% accuracy in identifying the state of the grasp. When deployed on

the robot, the *grasp reflex model* achieves one-shot tactile servoing on 6 novel tool instances with a success rate of about 73%. This success rate indicates that the *grasp reflex model* accurately captures grasping relationships for tools similar to the tool used for training, but may not sufficiently secure grasps on tools that are significantly different in size, shape, or weight from the training tool. We demonstrate that our proposed *grasp reflex model* is simple enough to be explainable while being reliable and generalizable enough to allow robots to improve attempted grasps and secure grasps in tool manipulation tasks.

3. **Human-Robot Red Teaming for Safety-Aware Reasoning** (Chapter 5) [98]: Towards tool-use and manipulation in safety-critical problem domains, we propose *human-robot red teaming* to allow humans and robots to explore the complexities of a problem domain, modify the team's modeled knowledge, and characterize the risks present in an environment. We formalize *safety-aware reasoning*—which includes: (a) hazard identification, (b) risk assessment, (c) risk mitigation, and (d) safety reporting—and demonstrate how the *human-robot red team* can improve the robot's ability to plan to complete tasks safely and perform risk mitigating actions. The *human-robot red team* aims to explore a complex problem domain, investigate hazards, and consider contingency risk mitigation strategies, by engaging in dialogue interactions to iteratively update the human-robot team's modeled knowledge. From the generated models, the robot can plan to complete tasks safely and perform other *safety-aware reasoning* tasks such as assessing risks and predicting the value or utility of performing different *risk mitigating actions* to improve task safety. We perform symbolic planning experiments to demonstrate that the robot can plan safe tasks in several diverse problem domains. We also perform robot execution experiments to demonstrate on two robots with different embodiments that the *human-robot red teamed* risk assessment models effectively assess and mitigate risks in different domains with

different definitions of safety. *Safety-aware reasoning* makes the robot an active collaborator in ensuring that tasks are executed safely, justifies the trust placed on robots, and allows robots to perform tool-use manipulation tasks alongside human users under varying safety conditions.

Chapter 2

Related Work

This dissertation aims to improve robot's abilities to autonomously plan and execute complex actions—such as those in assembly and tool-use tasks—while also reasoning over the safety of commanded tasks and ensuring robots can be trusted collaborators on human-robot teams. To better explain the novelty and significance of our contributions, we cover the related work within: Robot Programming Paradigms (Section 2.1); Object Affordances, Planning, and Action Execution (Section 2.2); Dexterous Robot Manipulation (Section 2.3); Robot Safety (Section 2.4); and Red Teaming (Section 2.5).

2.1 Robot Programming Paradigms

There have been several proposed robotics paradigms that describe how robots should be commanded to complete tasks. These paradigms also present a vision for how humans and robots should interact in complex tasks and share responsibility for the completion of these tasks.

Laird *et al.* [99] propose *interactive task learning (ITL)* to describe how robots can interactively learn to complete complex tasks. The authors envision a world in which robots and human operators interact similarly to students and instructors. In the ITL paradigm, the human operator (instructor) provides instructions for a task, the robot (student) asks questions to clarify instructions and discover information about the task, and in doing so, the robot learns to perform the task while learning about the task itself. ITL includes several key components: (1) **ease of interaction** where robots and humans should intuitively engage in dialogue about the task; (2) **task performance** allowing robots to perform complex tasks effectively; and (3) **dynamic scalability** of robot's learned knowledge to different tasks. Not only does ITL present a vision for the future of robotics and goals for human-robot interaction, but these ITL components provide a framework for conceptualizing how robotics research contributes to the ultimate goal of ITL. Throughout the remainder of the dissertation, we will emphasize these components—ease of interaction, task performance, and scalability or generalization—in the contributions of each work, specifically in the context of the CURED Framework as described in Section 1.1.2 and visualized in Figure 1.4.

Robot *Learning from Demonstration (LfD)* is an intuitive approach to robot programming that enables (non-expert) operators to program a robot through task demonstrations. In the context of the ITL key components, LfD supports effective task performance and ease of interaction [99]. However, knowledge learned through LfD does not necessarily scale well to novel tasks, as new tasks may require additional demonstrations [100]. *Semantic Robot Programming (SRP)* [101, 100] aims to improve the scalability of LfD by focusing on real-world tasks as opposed to simplified tasks. SRP allows human programmers to present the robot with an image of a task goal to command the robot to achieve that goal. The robot reasons over these goals and the current state of the world visually, creating a *scene graph* as an intermediate representation for current and target inter-object relations.

When SRP was first presented [101], the main contribution was to address perceptual challenges of recognizing cluttered scenes of real-world objects. Because the emphasis was on perception, the tasks were executed using atomic pick-and-place actions. We aim to build on the ease of interaction, task performance, and generalization of the SRP paradigm while planning over and executing more complex actions to complete task goals.

2.2 Object Affordances, Planning, and Action Execution

In safety-critical problem domains such as NASA's space exploration applications, we want robots to act in potentially dangerous, dynamic environments alongside human operators. For example, robots may perform tasks onboard the International Space Station (ISS) [91, 25], inside and outside of space vehicles and rovers [27, 28], and operating within supervised and shared autonomy schemes on human-robot teams [30, 31, 26]. We extend the planning and execution capabilities of robots operating in human environments to safety-critical space exploration tasks [3]. In particular, we explore how robots plan and execute complex tool-use manipulation tasks in safety-critical domains. To better understand our contributions, we cover related work in definitions of object affordances (Section 2.2.1), representations (Section 2.2.2) and perception (Section 2.2.3) of affordances, reasoning (Section 2.2.4) over affordances, and execution (Section 2.2.5) of robot actions.

2.2.1 Definitions of Object Affordances

Object affordances were originally presented by psychologist J. J. Gibson [33] to describe the actions that can be taken on an object by a particular agent in a particular environment. The concept of affordances has since been explored in many fields—including education [102], psychology, neuroscience, and robotics [103]—and the definition of affordances may vary between these fields [104]. Overall, the commonalities in definitions of *object affordances* highlight the importance of an agent's embodiment, knowledge, needs, and perceptions of the environment [105, 106, 107, 108]. For example, a heavy box may afford lifting to one person, but not another, depending on how much weight each person is capable of lifting.

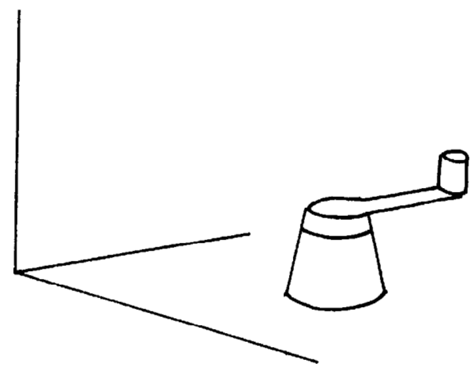
In the field of robotics, object affordances are important because the pick-and-place actions that may serve robots in factory assembly lines will not work for robots assisting with tasks in human environments. The pick-and-place abstraction oversimplifies the complex actions we expect robots to face in dynamic real-world domains. Instead, robots need to conceptualize, perceive, reason over, and execute many diverse actions and types of object interactions—such as pushing, pulling, flipping, and pouring, to name a few [109]. In this dissertation, we refer to *object affordances* as *action possibilities* on an object. Because the actions afforded by an object depend on the agent, it is important for robotic agents to perceive and infer the functions of objects [110], reason over affordance effects on the world, and execute complex actions to achieve task goals [111].

2.2.2 Affordance Representations

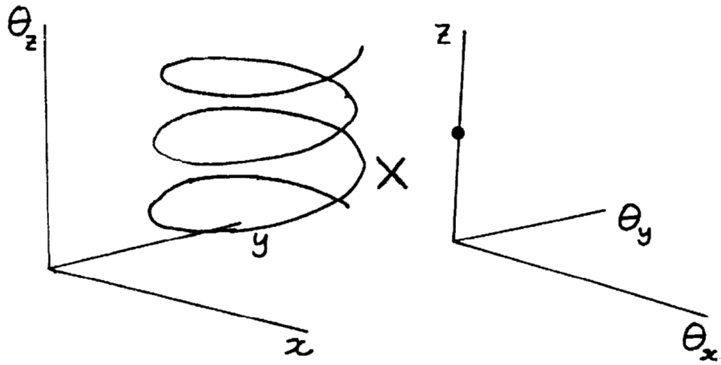
Though there are many proposed representations for object affordances [107], it is generally agreed that a robot's internal representation of affordances is grounded in the robot's behavior and the resulting effects on the environment [36]. Representations of affordances are important because these internal models become the basis on which robots will ground their actions, manipulation capabilities, and task goals [112]. For example, *object action complexes (OACs)* [113, 114, 115] are constructed by grounding the robot's sensorimotor processes—what the robot *senses* and how it *moves*—into symbolic actions that can be applied to future manipulation tasks. Affordance representations can capture important information such as: affordance and object symmetries [116]; conceptual relations between objects, attributes, and actions [117]; and spatial-temporal relations between agents, objects, and environments [118].

Zech *et al.* [107] present a detailed taxonomy and classification of robotic internal representations or computational models of affordances. While robots' models of object affordances have a number of properties, we highlight the affordance features most relevant to our work. Affordances can be perceived on different *levels*—such as feature-level, object-level, or environment-level—indicating how much the robot understands the context within which the affordance operates. The *order* of an affordance refers to how many affordances must be *chained* together for their combined effects to afford an action. For example, if a stack of two blocks is sitting on a table, the top block affords the pick-up-action as a *first-order* affordance because this action possibility is immediately available to the agent. The bottom block affords the pick-up-action up as a *second-order* affordance, since the agent must first move the top block before this action possibility becomes actionable. *Affordance chaining* describes how affordances can be sequenced (such as actions in a task plan), overlapped (for example, two behaviors being performed simultaneously in bimanual manipulation), or work together concurrently (by acting on direct and indirect objects such as screwdriver and screw). *Selective attention* to affordances allows the robot to focus on the affordances most relevant to a particular task goal. *Prediction* allows robots to identify meaningful affordances, by classifying sensory inputs to appropriate actions or predicting the likelihood an outcome is achieved. Affordance knowledge should *generalize* to new situations, meaning the robot can transfer its understanding of an affordance to multiple tasks with minimal modifications. Through our work, we aim to recognize affordances on multiple levels and orders, chain affordances together by predicting their effects, and generalize affordance knowledge to novel action instances.

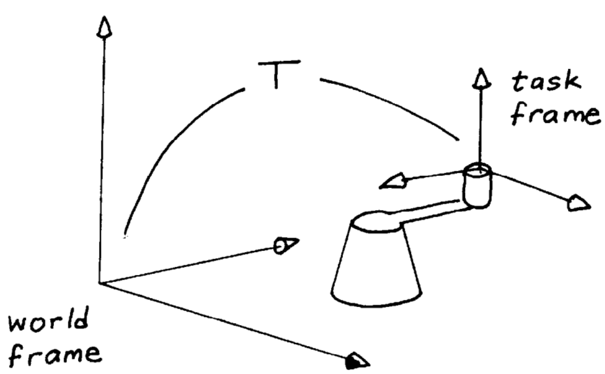
In this dissertation, we focus on representations of affordances as *object-centric behaviors* [5, 119] in order to emphasize the importance of reasoning over and acting on the object itself. Object-centric behaviors are expressed in *task frames* that simplify action expression. Figure 2.1 replicates an example from Ballard [5] demonstrating the power of task frames. For example, expressing a crank-turning-action in world frame involves reasoning over the arc the crank might follow (as seen in Figure 2.1b). But expressing the same crank-turning-action in a task frame fixed to the crank handle involves applying force along an axis of the crank handle



(a) The goal of the task is to turn the crank by performing a crank-turning-action.



(b) The crank-turning-action represented in world frame. This action involves rotating the crank handle around its vertical z-axis, which causes x- and y-coordinate translational changes, as seen on the left. Meanwhile, there are no rotational changes in the x- and y-directions and there is no position offset in the z-direction, as seen on the right.



(c) The crank-turning-action represented in a *task frame* attached to the crank handle. Since the task frame moves with the crank handle, representing the crank-turning-action in the task frame simplifies how the action is expressed. The task frame indicates that turning the crank involves applying force along an axis relative to the crank handle, namely the axis orthogonal to both the crank handle and crank axle.

Figure 2.1: Example from Ballard [5] showing how representing actions in *task frames* as opposed to world frames can simplify the expression of the action.

(visualized in Figure 2.1c) until resistance is met [5]. Expressing behaviors relative to the object also means execution can be thought of as pose invariant manipulation policies [120], since no matter what the object's pose is in the world frame, the action relative to that object remains the same.

A prominent representation of affordances is as templates [121] that describe general behaviors relative to an object. *Affordance templates* [122, 123] represent actions as sequences of end-effector waypoints and end-effector poses the robot must move through to achieve some goal. Object-centric behavior templates can be thought of as actions parameterized relative to an object, where action parameters may be changed based on observations about a particular object [124] or category of objects [125]. Templates are appealing representations because they capture complex behaviors as a sequence of relatively sparse waypoints and they can be flexibly modified depending on the particular action instance. Affordances can also be represented in terms of primitive manipulation behaviors [126]. For example, *affordance primitives* [127, 128] have been shown to represent complex behaviors, such as arc or screwing motions.

In our work, we represent actions as affordance templates [122, 123] combined with object-centric controllers for more complex motions, as described in Section 2.3.2.

2.2.3 Affordance Perception

Before planning over affordances or executing the planned actions, a robot must be able to perceive the affordances in a scene. *Affordance-based perception* focuses on allowing the robot to detect, label, and recognize what actions are afforded by certain objects, placing the emphasis on perceiving the affordances themselves rather than the objects. Affordance-based perception affects the robot's understanding of the scene [129, 130]. The importance of perceiving affordances has been demonstrated in many use cases, including: visually recognizing affordances for categories of objects [131]; semantically labeling affordances [132]; using a knowledge graph to perform visual reasoning and visual inference tasks [117]; detecting affordances relative to graspable parts of tools [38]; or using deep learning approaches to predict functions of objects based on visual features [133].

Research on affordance perception demonstrates the close relationship between perceiving and acting, showing that one can often be used to improve the other. Taking actions to move and clear away objects can help reduce perceptual uncertainty in cluttered environments [134]. Real-time perception can also improve reactivity when performing actions [135] by providing live feedback on dynamic environment changes. Research in *active perception* [136, 137] and *interactive perception* [138, 139] demonstrates that interacting with objects can improve segmentation and detection of objects, object sub-parts, and their affordances.

Recognition and use of an affordance can depend on how the robot perceives the affordance. In particular for object-centric affordances, the robot needs to choose an appropriate coordinate frame for representing the action and attach the frame to an object or object part. We utilize *Affordance Coordinate Frames (ACFs)* [140, 141], which attach affordances to particular object features as localized coordinate frames—and therefore coincide with the object-centric affordance representations used throughout our work. ACFs demonstrate the importance of attaching affordances to particular sub-parts of objects, also referred to as parts-based object reasoning [142] and parts-based affordance perception [143]. Perceiving objects as a collection of sub-parts is particularly important for articulated objects with moving parts.

Some work in this dissertation assumes automatic localization of ACFs, while other work relies on human-in-the-loop pose estimation [144] or human-in-the-loop affordance registration [145, 146].

2.2.4 Affordance Reasoning and Learning

Reasoning over and learning about affordances is crucial to the robot's understanding of how to apply object affordances to real-world task goals. In particular, the robot must understand how an affordance embodies the close relationship between objects, actions, and effects [147, 148]. Some research demonstrating the power of affordance reasoning includes inferring categories of laundry from visual and geometric features to perform folding tasks [125] or reasoning over a knowledge graph encoding object, attribute, and affordance information to perform inference tasks [117].

Reasoning can significantly impact a robot's understanding of affordances. For example, Moratz *et al.* [149] demonstrate that when considering object affordances and functional features, a robot can reason over the agent perspective to understand a human user's spatial understanding of a task. Koppula *et al.* [118] show that affordances can be reasoned over as spatial-temporal relations that allow assistive robots to anticipate human users' intentions and react with the corresponding assistive action accordingly. Yang *et al.* [150] explore how robots can understand the syntactical and semantic meanings of an object affordance by observing humans performing actions. This research collectively emphasizes the relationship between affordance understanding and affordance reasoning.

Recent work has taken interest in tool construction tasks. Constructing new tools by comparing the geometric properties of available tool parts to the geometry of a reference tool [35, 13] demonstrates advanced reasoning over objects, their affordances, and how an object can be used to achieve a desired effect. Nair *et al.* [13] formalize the multiple levels a robot can reason over to construct a tool that achieves a specific task. *Object equivalence solutions* construct a new (equivalent) tool that affords the same actions and achieves the same effect as the reference tool. *Object-action equivalence solutions* construct a new tool that affords different actions but achieves the same effect as the reference tool. *Object-action-effect equivalence solutions* construct

Task Goal: Attach Two Boards

	Reference Solution	Equivalence Solution		
		O	**OA**	**OAE**
Object	Screwdriver	Knife Blade	Hammer	Rope
Action	Turn	—	Pound	Tie
Effect	Boards Screwed	—	—	Boards Tied

Table 2.1: Example *object* (O), *object-action*, (OA), and *object-action-effect* (OAE) equivalence solutions for a tool construction task from Nair *et al.* [13], adapted with permission of the author. To attach two wooden boards, the reference solution uses a screwdriver (object) to turn (action) a screw and tighten (effect) the screw. Each equivalence solution considers different tool objects, actions, or effects to achieve the task goal. The symbol "—" indicates that the equivalence solution preserves the action or effect of the reference solution.

a new tool that affords different actions with different effects from the reference tool, but results in an equivalent achievement of the task. Table 2.1 describes a tool construction example from Nair *et al.* [13], showing the different levels of equivalent solutions[1] to achieve the task goal of attaching two pieces of wood. Note that *object* and *object-action* equivalence solutions constrain the robot's planning by preserving properties of the given reference solution. In contrast, *object-action-effect* equivalence solutions do not constrain the robot's planning relative to the reference solution. This means that finding an *object-action-effect* equivalence solution can be achieved through task planning, since the robot must achieve the task goal without any consideration for a reference solution. Tool construction tasks demonstrate the power of reasoning over the objects, actions, and effects encapsulated by object affordances.

An important sub-problem within affordance reasoning research is how robots can learn about affordances. Many works explore learning about affordances from experience, such as: learning to identify affordances in clutter [151]; learning probabilistic affordance models [147, 148]; learning to complete pick-and-place tasks using deep reinforcement learning [152]; learning what affordances are most likely to successfully

[1]Broadly, equivalence solutions would also include *action* and *effect* equivalences. Considering the example in Table 2.1, an action equivalence solution may use the reference screwdriver object to *pound* the screw into place. The work presented by Nair *et al.* [13] focuses on reasoning over tool construction or substitution tasks, which is why their example focuses on the equivalence solutions that involve variations on object equivalence.

achieve their effects using relationships between sensorimotor processes and task success [153]; and learning affordance concepts (such as affordance categories or effects) through experience and exploration of the world [154, 155]. Hart *et al.* [156] learn to perform complex manipulation by rewarding exploratory behaviors that generate novel sensorimotor feedback. This research demonstrates the importance of learning about affordances from experience and exploring the space of actions, effects, and sensorimotor inputs.

Other works on affordance learning show the significance of reasoning over affordances involving multiple objects. Multi-object affordances can be learned in a number of ways. Moldovan *et al.* [157] learn about relational actions—actions that involve the (spatial) relationship between two or more objects—by observing the effects of these actions. Ugur *et al.* [158] use knowledge of single-object affordances to guide learning about multi-object affordances. Similarly, Fichtl *et al.* [159] leverage past knowledge through bootstrapping or transfer learning. By modifying their understanding of actions as they are executed in real-world tasks, robots can effectively achieve lifelong learning [160] and continuously improve their affordance reasoning capabilities.

We explore how robots use policies to plan multi-object affordances in tool-use and construction tasks. Because tool-use and assembly tasks require long-horizon planning and in-depth understanding of an object's affordances, these tasks present several unique challenges for action planning as discussed further in Section 2.3.1.

2.2.5 Execution of Afforded Actions

After perceiving and planning over affordances in a scene, the robot selects one of the available "possibilities for action" and must be able to execute the (often complex) action. The choice of affordance representation often lends itself naturally to the execution of that afforded action—for example, object action complexes [113, 114, 115], object templates [121], and affordance templates [122, 123]—and affordances become a basis for robot control [161]. Affordance templates are powerful affordance representations that lend themselves to execution of the afforded action since they encode an action as a sequence of end-effector waypoints. However, affordance templates do not provide any description of how the robot should move between waypoints. *Affordance wayfields* [162] aim to provide direction for action execution by describing actions as potential functions defining sets of trajectories that will achieve the desired action effect. Robot actions and execution of afforded actions can also provide the robot with additional information to learn about the object and its functions [163].

In this dissertation, we combine affordance templates [122, 123] and object-centric controllers to command the robot to perform complex assembly and tool-use actions. The challenges of tool-use and assembly tasks are explored in Section 2.3.1. Section 2.3.2 describes execution of robot behaviors using object-centric controllers.

2.3 Dexterous Robot Manipulation

NASA has deployed or plans to deploy assistive robots in a variety of domains in order to assist humans in space exploration tasks such as maintenance, hand-off, cleaning, and equipment management tasks [91, 30, 31, 26, 3, 25, 27, 28]. Many of these tasks require the ability to use tools, advanced reasoning over actions, and execution of complex manipulation behaviors. In this dissertation, we aim to improve the planning and execution capabilities of robots so they can achieve dexterous manipulation in safety-critical space exploration tasks. We cover related work motivating our interest in tool-use and assembly tasks (Section 2.3.1), executing complex behaviors (Section 2.3.2), and grasping tools intentionally to perform subsequent manipulation tasks (Section 2.3.3).

2.3.1 Tool-Use and Assembly Tasks

Tool-use and assembly tasks have gained much interest for developing dexterous robot manipulation capabilities [164, 165, 166, 167]. There is even interest for performing robotic tool-use and assembly tasks in space [168, 169, 170]. These tasks present a wide variety of challenges. For example, challenges of tool-use and assembly tasks include: perceiving objects in clutter; detecting and predicting articulated parts-based affordances; reasoning over affordances involving multiple (direct and indirect) objects; planning to execute actions that may achieve multiple effects; sequencing behaviors appropriately; concurrently executing behaviors to achieve sub-goals simultaneously; and performing complex manipulation trajectories. For these reasons, tool-use and assembly tasks require advanced robotic capabilities, especially in terms of planning actions using policies that select the next action and executing afforded actions.

Research has concluded that tool construction and tool-use tasks require advanced causal reasoning. As a result, agents that successfully reason over cause and effect and use tools to achieve task goals have high levels of intelligence [34]. Stoytchev *et al.* [36] emphasize the importance of behavior-grounded affordance understanding, specifically for tool affordances that involve complex behaviors and multiple object interactions. Several works explore how learning from demonstration approaches can be used to teach robots manipulation primitives required to perform assembly tasks [171, 172, 173]. Lee *et al.* [9, 10] developed the IKEA Furniture Assembly Environment as a testbed for perception, planning, and manipulation capabilities required to perform construction tasks. Nair *et al.* [35, 13] use geometric reasoning to compare tool components to a reference tool in order to construct tools to achieve a given task goal. Taken together, these works emphasize the complexity of these tasks and the interest in addressing the challenges of tool-use and assembly tasks. With this motivation, our work aims to improve robot capabilities in multi-object affordance reasoning and complex action execution to address challenges of tool-use and assembly tasks.

2.3.2 Multi-Objective Behaviors and Control Basis

As discussed in Section 2.2, we are interested in representing object affordances as object-centric behaviors [5]. We execute these object-centric behaviors as object-centric controllers, which send joint commands such that the robot performs low-level motion primitives. Controllers can be used in place of or alongside traditional motion planners [174, 175, 176, 177, 178] because they offer additional reactivity to dynamic changes in the environment [179, 180, 181]. In fact, motion planning and robot control are often closely related fields of research. Motion planning research emphasizes the importance of probabilistic inference [182] and creating trajectories as combinations of manipulation paths [183]. Planning tasks can also be hierarchically decomposed so that robots can plan simpler sub-problems [184] or robots can construct kinematic solutions to task-level problems from solutions to simpler problems [185]. Task and motion planning (TAMP) demonstrates the close relationship between task planning, motion planning, and trajectory execution by using object-centric behaviors to adapt to changing environments at runtime [119]. These same principles of hierarchical decomposition into simpler sub-problems and reactivity to dynamic changes are crucial to object-centric controllers.

Object-centric controllers can be used as part of a *control basis*, or a set of controllers that serves as the building blocks for robot behavior [45, 46, 47]. Controllers can be composed through *nullspace composition* such that multiple objectives can be achieved concurrently [186]. For example, suppose we want the robot to grasp an object using two low-level controllers: controller ϕ_{pos} positions the robot's end-effector close to the object and controller ϕ_{align} aligns the end-effector's approach axis with the object. Composing these controllers enforces a priority using the "subject-to"

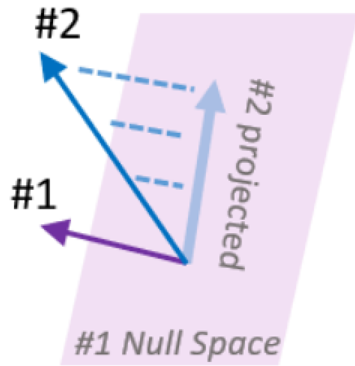


Figure 2.2: Visualization of *nullspace composition* operation from Sharma *et al.* [6], reprinted with permission of the author. Controller #1 is higher priority than Controller #2 ($\phi_2 \lhd \phi_1$ or equivalently, Controller #2 subject-to Controller #1). To avoid disturbing progress made toward achieving the objective of Controller #1, the commands of Controller #2 are projected into the nullspace of the higher priority command. Nullspace projection ensures that the lower priority controller commands are tangential to the higher priority commands.

relation $\lhd$. For example, composition:

$$\phi_{\text{align}} \lhd \phi_{\text{pos}} \tag{2.1}$$

(read "alignment subject-to position") indicates that the positioning behavior is highest priority. Mathematically, this means the alignment controller's joint commands will be projected into the nullspace of the position controller, as visualized in Figure 2.2 (from Sharma *et al.* [6]). Practically, this means that any progress made towards achieving the alignment controller's objective will not disturb or disrupt any progress made by the positioning controller. Figure 2.3 (from Sharma *et al.* [6]) provides an example of how nullspace composition results in a composed controller command.

Composing controllers is a powerful tool for robot manipulation since complex behaviors can be executed through compositions of simple control behaviors. These controllers are often implemented using potential functions [40]. There has been much work defining potential functions for standard robot manipulation tasks [43], such as repulsive potential fields for obstacle avoidance [187, 42], control laws with constraints

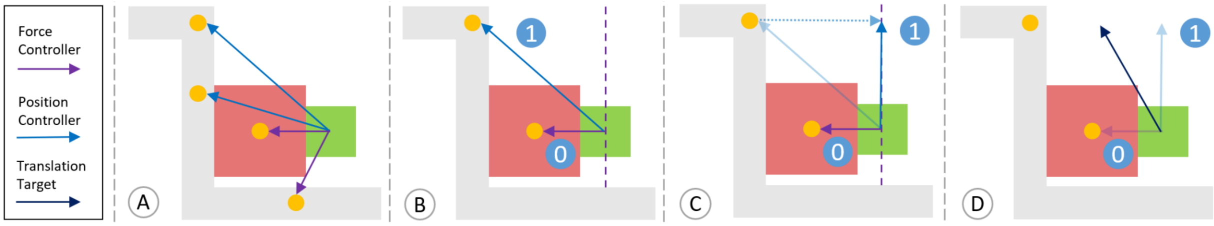


Figure 2.3: Example controller composition from Sharma *et al.* [6], reprinted with permission of the author. The robot's end-effector (the green block) must push the red block up vertically along the wall. (A) The robot can choose different position and force controller objectives. (B) The robot chooses and prioritizes two objectives. The force controller (labeled 0) is higher priority than the position controller (labeled 1), indicating the importance of keeping the red block pressed against the wall. (C) To compose the controllers, the lower priority controller command is projected into the nullspace of the higher priority command (represented by the purple dashed line). The nullspace composition operation ensures that the lower priority controller command operates tangentially to the higher priority command, preventing it from interfering with any progress made toward the higher priority objective. (D) The projected controller commands can be combined to find the command for the composed multi-objective behavior.

and contact requirements [44], and conditioning behaviors such as avoiding joint limits or singularities [48, 49, 50]. With this toolbox of potential functions serving as behavior building blocks in a control basis, a wide variety of complex robot manipulation tasks can be performed. For example, multi-objective behaviors through controller compositions have been used to perform tasks such as grasping [45, 46, 47] and composed bimanual manipulation tasks that require relative control between end-effectors [188, 189, 190] as in a writing task [191]. Taken together, research on compositions of controllers within a control basis demonstrates the wide variety of robot tasks that can be performed.

Controller compositions lend themselves naturally to hierarchical task decomposition. Many works explore the relationships between sequences and compositions of behaviors, such as when actions must be performed sequentially or concurrently [186] and how to sequence behaviors through basins of attraction [7, 41] to achieve composed effects. A visualization of how sequential composition of robot behaviors can achieve

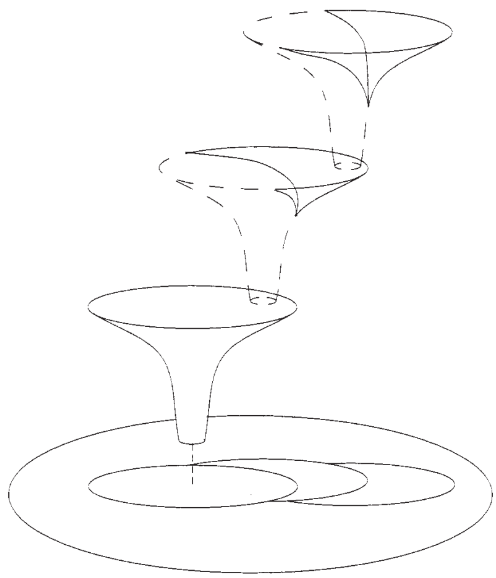


Figure 2.4: Sequential composition of robot behaviors from Burridge *et al.* [7], reprinted with permission of the author. Behaviors are visualized as funnels, where the controller objective lies in the funnel's basin of attraction. When a higher level controller falls into the basin of attraction for a lower level controller, the next controller takes over [7]. By sequencing compositions of behaviors, the robot can achieve the *composed effects* of all controllers.

composed effects can be seen in Figure 2.4 (from Burridge *et al.* [7]). This research emphasizes the power of hierarchical reasoning in manipulation tasks [49, 50], where complex actions can be decomposed into simpler control laws. Composed controllers build on research involving hierarchical reasoning, planning, and control [192, 32] and subsumption architectures [193, 194, 195] that prioritize behaviors over others. The composition of simple control basis behaviors to create multi-objective behaviors parallels the hierarchical composition of sub-problem solutions to achieve more complex tasks.

Controller compositions are powerful because they can be applied to many manipulation tasks. However, composing controllers requires a prioritization between behaviors such as in Equation 2.1. These behavior priorities are typically predefined and hard-coded [45, 46, 47, 48, 49, 50], often based on experimental or qualitative observations from expert users rather than concrete quantitative reasoning [6]. While it may be useful for (expert) human users to provide qualitative insights to robots as an initial hypothesis or to guide the robot's understanding of a task, we ultimately want to minimize expert knowledge engineering. We expect robots to consider qualitative insights from humans while also learning from their own experiences and reasoning over tasks autonomously. This is particularly important for promoting ease of interaction for non-expert users who may not have a deep understanding of robot control laws and for promoting explainable robot actions in safety-critical problem domains. Recent work has explored how robots can learn to autonomously compose controllers in atomic actions such as pushing and turning [6]. In Chapter 3, we extend this work by exploring how robots can autonomously compose controllers in long-horizon construction tasks by quantitatively reasoning over the desired composed effects of the controller behaviors.

2.3.3 Grasp Behavior Modeling

Before a robot can execute multi-objective tool-use and assembly tasks, the robot must be able to intentionally grasp the tool. Robot grasping is highly dependent on the hardware of the gripper [196, 197]. Grasping restrains or manipulates an object by achieving contacts [198] and applying forces to the object [199]. Robots may need to perform grasping tasks in unstructured environments [200] or as part of bimanual manipulation tasks [201]. Grasping tools is particularly challenging. Tool-use tasks require intentional grasps—also called manipulation-oriented grasps [37], meaningful semantic task grasps [202], or task-specific grasps [38]—such that the tool can be used within the subsequent manipulation task [39].

Many researchers have explored various approaches to robot grasping [203, 204, 166]. Robot grasping can be performed autonomously, with real-time teleoperation from a human operator, or with autonomous assistance to the operator [205]. Grasped objects can be modeled geometrically or approximated using geometric primitives [206]. Other works avoid modeling object geometries by representing specific objects through generalized [207] or canonical shapes [208]. Grasps can be represented as grasp primitives [209] or primitive hand postures [210]. Grasping strategies can be learned [211] through reinforcement learning [212, 213, 214, 215], deep learning approaches for detecting task-specific grasping affordances [38], deep learning for identifying semantic task grasps on household objects [202], or deep reinforcement learning [216, 217, 218] over experiences performing pick-and-place tasks [152]. Some works combine analytic modeling and learning into a hybrid approach [219]. Probabilistic approaches allow for generation and probabilistic ranking of candidate grasp poses [220, 221] or imitation of learned grasp poses to be transferred to other objects [222]. Others explore biologically-inspired grasping algorithms through how the grasps are learned [223, 209], represented [224], or performed [225], even taking inspiration from human grasping [226, 227, 228].

The robot should be able to validate that its grasps on objects are secure through some form of sensory feedback [229]. Inspired by the desire to utilize sensory feedback, *tactile servoing* methods—a counterpart to visual servoing—use data from tactile or contact sensors to control a robot [230]. These approaches often require large arrays of tactile contact sensors [231] or multi-fingered robot end-effectors [232] in order to utilize abundant sensory signals. Tactile servoing can be used to achieve a variety of robot control tasks [233, 234] and can be used in conjunction with visual servoing to improve perception or control tasks [229, 137, 235, 236].

Due to the amount of data provided by arrays of tactile contact sensors, tactile servoing often uses data-driven machine learning approaches such as convolutional neural networks [237] or deep learning neural networks [238]. Others use offline neural network based learning to augment data-driven learning from demonstration [239]. While deep learning approaches have proven effective, these black-box models reduce explainability, which can reduce trust [62, 60, 61] and prove to be problematic for safety-critical problem domains [71, 72, 73, 74]. In Chapter 4, we take inspiration from data-driven learning from demonstration approaches [239] and aim to achieve tactile servoing using a more explainable *grasp reflex model* for tool-use grasps in safety-critical applications.

2.4 Robot Safety

The focus of the work presented in this dissertation is how robots can operate in safety-critical domains alongside humans [91, 26, 3, 27, 28]. In this section, we review related work crucial to promoting safety and trust, specifically the importance of intuitive natural language communication and dialogue between humans and robots (Section 2.4.1), organizational and robotics safety (Section 2.4.2), and the role of trust in cooperative safety tasks (Section 2.4.3).

2.4.1 Natural Language Processing

Researchers have long investigated different ways for users to intuitively interact with and program robots. Different interaction modalities provide abundant input signals, such as gestures or facial expressions [240, 241], eye-tracking [242, 243], a point-and-click interface in the real world [244], virtual reality [245, 246], or learning from demonstration [247, 248, 249, 171]. In particular, researchers have explored how to use language to intuitively communicate task goals to robots, especially by non-expert users who do not have programming or robotics experience [250, 251]. In the 1970s, the system SHRDLU [252, 253, 254, 255, 256, 257] was developed, which carried out natural language commands in a virtual environment. Since then, researchers have aimed to expand the use of natural language to command intelligent agents and robots [258, 259].

Many works demonstrate the power of commanding robots through natural language processing (NLP). Dzifcak *et al.* [260] explore how to translate natural language instructions into descriptions of task goals and actions. Chernova *et al.* [261] use data-mining for robots to ground action-oriented natural language. Tellex *et al.* [262] present Generalized Grounding Graphs for probabilistically inferring the sequence of actions required to execute a command. Matuszek *et al.* [263] investigate how robots can learn what objects are being referred to in deictic gestures and language (gestures and language that draw attention to objects without naming them directly). Several works explore how robots can ground abstract spatial concepts (such as relative relationships between objects) to execute natural language commands [264, 265, 266, 267]. Many works explore understanding natural language in route navigation tasks [268, 269, 270, 271], including commands involving verbs that imply motion [272], commands that imply navigation constraints [273], and commands that imply environment information [274, 275]. Google's SayCan [276] combines a large language model with affordance knowledge to allow robots to reason over natural language in long-horizon tasks. IBM's CIMON [277, 278, 279] is an assistive voice-commanded robot onboard the International Space Station (ISS). The presence of voice interfaces—such as Amazon's Alexa, Apple's Siri, Google's Assistant, and Microsoft's Cortana [280, 281, 282]—in everyday life demonstrate that restricted language may allow users to achieve similar or better task performance than natural (unrestricted) language without detracting from overall user experience [283]. These works demonstrate widespread interest in using (natural or restricted) language to command robots and the challenges of grounding natural language in robot understanding.

Since human natural language tends to include ambiguity, an important component of NLP research is grounding abstract language commands into concrete robot actions. Antunes *et al.* [284] state that "verbal instructions do not have a one-to-one mapping to robot actions." Therefore, it is important to have a middle-ground between natural language commands and robot action where the robot conceptualizes and represents the commanded task goals. These middle-ground models may take the form of: internal world models that enable visualization or "imagination" of task plans [285]; mental models where a robot reasons over other agents' perspectives of the commanded task [286]; planning over intermediate goal states rather than step-by-step operations to complete the commanded task [287]; or probabilistic planners reasoning over symbolic representations of task goals [284].

In this dissertation, we use semantic frames to represent language commands and serve as a middle-ground between command and robot action. *Semantic frames* are used in NLP to represent a scene being acted out [288, 289, 290, 291]. FrameNet [288] emphasizes that a verb alone is not sufficient to describe a scene or action, and *frame elements* are necessary to describe agents and direct and indirect objects involved in the action. For example, the verb "give" cannot be acted out until we know what object is being given, to whom, and by whom. FrameNet uses hand-annotated *lexical units* to map language into the appropriate semantic frame by expressing how frame elements relate to a command. RoboFrameNet [292] extends FrameNet [288] by using semantic frames as a middle-ground between spoken commands and robot action. Semantic frames are of interest for robotics applications because representations of object affordances generally do not explicitly note the direct and indirect objects being acted on,[2] which limits the complexity of robot action that can be performed [107]. In contrast, semantic frames augment robot understanding of the action being performed by describing the objects being acted on. RoboFrameNet [292] explores how to interpret spoken commands as implied affordances or potential actions, and ground any unbound variables into concrete

[2]It is important to distinguish between the *concept* of affordances and *computational models* or *representations* of affordances. The *concept* of an affordance as a broad "opportunity for action" or "action possibility" does entail requirements such as direct and indirect objects. However, in practice, the *computational models* of affordances may not include these direct and indirect objects [107]. This implies that affordance representations should be enriched, and some works use semantic frames as a computational model of affordances [292, 293]. One advantage of using semantic frames as affordance representations is that binding language elements into the frame is related to constraining and binding variables to execute a grounded action.

robot actions through task planning. RoboFrameNet interprets spoken commands as text, then parses the text [294] to instantiate a semantic frame. The semantic frame contains relevant information about the task such as agents involved, direct and indirect objects to be acted on, and the sequence of actions required to carry out the commanded task [293].

It is important to note that while this dissertation uses natural language commands because they serve as an intuitive interface for non-expert users to communicate task goals to robots as a component of the CURED Framework, this dissertation does not contribute to active NLP research. As a result, we limit the capabilities of our language processing as some capabilities will be outside the scope of our work. In Chapter 5, we address how robots can *safely reason over* and *safely execute* commands and we demonstrate the importance of robots engaging in dialogue with human operators about safety. We take inspiration from DOROTHIE [11] and implement an "English-like" interface in the form of a dialogue decision tree, rather than a complete natural language interface. In our implementation, safety-aware dialogue takes the form of some (limited) communication acts the robot can engage in with the human operators. Since the contribution of this work is *safety-aware reasoning* and not NLP, some natural language capabilities are outside the scope of this project. We leave it to future work to extend our methods to process more abundant natural language commands and process more robust safety-aware dialogue.

2.4.2 Safety and Language in Robotics Applications

As described in Section 2.4.1, there are many applications for using natural language to command robots. However, many question whether large language models truly lend themselves to deep understanding of the tasks to which they are being applied [295]. Furthermore, in safety-critical applications, the potential for anything and everything non-expert users say to trigger a robot action does not necessarily result in a safe and user-friendly plan of action. Some works that apply large language models even acknowledge that safety of the commanded agent's movements and the effect of the agent's action on user experience are not currently addressed in their work [51]. This indicates that current work on natural language in robotics applications is limited in its use in safety-critical domains.

Despite the fact that safety has not been addressed in works that control robots using language, much research emphasizes the importance of safety when robots are working alongside humans [296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 64, 53, 54, 57, 55, 307, 56], especially in collaborative and assistive tasks [308, 306]. There is clearly a need for robot systems that can reason over language in a safety-aware manner. We can take inspiration from government and industry safety cultures and risk assessment standards to provide a principled way for robots to reason over safety [71, 72, 73, 74].

As an example risk assessment standard, a *risk assessment matrix* is a commonly used tool for organizations to quantitatively evaluate the likelihood and consequences of undesirable events or hazards [75, 309, 310]. Table 2.2 shows an example risk assessment matrix, similar to those presented in the literature [309, 310]. These risk assessment matrices aim to help organizations assign a probability or likelihood value L and severity of consequence value C to compute a risk score $R = L \cdot C$. Intuitively, undesirable events that are more likely to occur and/or that have more severe consequences result in a higher risk score.

Once we have assessed the level of risk, we must take action to mitigate the hazard. NASA's safety culture emphasizes the importance of risk assessment and risk reduction throughout the entire life-cycle of a project, rather than an isolated review at the beginning or end of the project [71, 72, 73, 74]. When hazards do arise, NASA's safety program outlines a *risk reduction protocol* [76] with prioritized actions a team should take to address hazards. In order of preference, NASA's risk

		Consequence				
		1	2	3	4	5
Likelihood	5	5	10	15	20	25
	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5

Table 2.2: Example risk assessment matrix. Likelihood values indicate the probability of a hazard occurring. In this case, values range from $[1, 5]$ in order of increasing probability (*improbable*, *remote*, *occasional*, *probable*, and *frequent*, respectively). Consequence values indicate the severity of a hazard occurring. Values range from $[1, 5]$ in order of increasing severity (*insignificant*, *minor*, *moderate*, *major*, and *severe*, respectively). The risk score R is computed from the likelihood L and consequence C as $R = L \cdot C$. Risk scores can be bucketed into more interpretable categories to indicate different levels of risk. In this example, each matrix element is color-coded (blue, green, yellow, orange, red) to indicate the corresponding risk level (*negligible*, *low*, *medium*, *high*, and *extreme*, respectively).

reduction protocol indicates the team should consider the following risk reduction actions:

1) **Design**: prevent the hazard or minimize the risk through improved design or operation.

2) **Guards**: if design or operation cannot be changed, incorporate safety devices or guards to minimize the risk.

3) **Warnings**: in the event that safety guards fail or cannot be implemented, provide cautions or warning devices such as alarms.

4) **Training**: develop administrative procedures and training to reduce risk.

5) **Accept**: after all other actions have been considered or taken, accept the residual risks.

Note that the earlier risk reduction actions should be prioritized, as they can help prevent hazards entirely, while later risk reduction actions tend to accept that the hazardous event will happen and try to work around these hazards to minimize negative impact. This example risk reduction protocol demonstrates the importance of considering multiple strategies to reduce or mitigate every hazard that may occur during operations.

Taken together, these organizational practices from government and industry safety and risk assessment standards give us inspiration for how we expect robots to perform *safety-aware reasoning*. We expect robots to understand hazardous events in the problem domain, quantitatively assess risks according to the likelihood and consequences of undesirable events [75, 309, 310], and take appropriate risk reduction actions where possible [76]. Safe robot operations require the robot to have a sense of spatial awareness [311, 312, 313] and understand the physical safety concerns when working in close proximity to humans [52, 53, 54, 55, 56, 57, 297, 298, 299, 300, 301]. In Chapter 5, we explore how robots can use these principles to safely execute different tasks.

2.4.3 Trust and Cooperation

We aim to allow humans and robots to engage in a dialogue about safety in collaborative tasks. *Cooperative* tasks—in which agents work together to achieve positive-sum "win-win" outcomes—involve vulnerability [61]. The social nature of cooperative tasks [58, 59] makes *trust* between robot and human user/operator crucial [60]. Inspired by work on effective communication in cooperative conversations [314, 315, 316, 317, 318, 319], we expect robots to participate in the "social exchange relationship" often associated with interpersonal trust [58].

When reasoning over and making decisions about tasks, agents utilize simplifying models of the world. Even for human agents, reliance on simplifying models[3] is necessary when exhaustive evaluation of alternative options is impractical [60, 58].

[3]For example, engineers use simplifying mathematical models—which represent idealized limited situations—to approximate real physical systems [320].

Trust plays an important role in decision-making because it can help further reduce complexity and uncertainty, and can allow agents to focus on relevant components of the task by trusting the other agents will cooperate [60]. However, this trust must be properly *earned* in order to demonstrate that the agent can play a role in future cooperative tasks. The Ethics, Trust, Cooperation (ETC) Framework [61] explains that to earn trust, robots must show that they can act as "*members* of our society" by operating according to the social norms, morality, and ethics of the society within which they function [62]. Kuipers [63] observes that viewing robots as participants in our society follows similar trends in recognizing the very real and active role other embodied or disembodied entities play:

> Autonomous vehicles must be trusted to behave safely and ethically in both routine traffic and emergency situations. Other AIs that are not physically embodied, such as high-speed trading systems or social networks, should also behave safely and ethically. Large-scale institutions can also be considered as intelligent entities: for-profit and nonprofit corporations, governments, churches, unions, and other corporate entities. For all of these entities participating in society, the function of ethics is the same—to encourage positive-sum interactions and discourage negative-sum ones, supporting the survival and thriving of society as a whole.

We want robots and humans to engage in dialogue about safety. In order to function in this cooperative task, robots must earn the trust of their human operators and function as active participants in the cooperative task of minimizing risks while operating in safety-critical domains.

Robots and automation acting in cooperative tasks are often not relied on appropriately, specifically when reliance on the system does not match the robot's true capabilities. More specifically, if a human's trust exceeds the system's capabilities, they *overtrust* the system, which can lead to misuse of automation. If a human's trust falls short of the system's capabilities, then they *mistrust* the system, which can lead to disuse of automation and failure to obtain the available benefits of the technology. Our goal is for trust between robots and human operators to be well *calibrated*, in which trust appropriately matches the system's capabilities and the automation is properly used [58, 59].

In order to properly calibrate trust between human and automated agents, Lee and See [58] describe principles for designing trustable technology. Appropriate trust and reliance on automation requires that *robot capabilities be clearly communicated to the human users*. Promoting better understanding of robots' capabilities can be achieved by revealing how the automation operates [58] and relying on simple decision-making algorithms [69, 70]. Appropriate trust depends on how well the human operator understands that the *context, circumstances, and environment* within which the robot operates *affect its capabilities and performance* [58]. For example, it may be necessary to make changes to human environments to improve robot capabilities. However, environmental changes that improve robot operation will also very likely simplify human action in those environments [27, 28]. Appropriate trust and reliance not only depend on individual human operators' interactions with the robot, but on broader organizational and cultural factors [58]. For this reason, we take inspiration from organizational safety culture, risk assessment, and risk mitigation strategies in industry [309, 310] and government [71, 72, 73, 74, 75, 76]. These organizational practices emphasize the importance of *safety-aware reasoning* and risk mitigation at all stages of the life-cycle of a project. We expect robots to operate not just as members of society, but participants within a broader safety culture.

One key component of promoting trust between humans and robots is clear explanations of robot actions. Robots taking actions that cannot be meaningfully explained will cause users to distrust the robot [321, 58, 59] as unreliable robots can also be unsafe [64]. Eroded trust and a reputation for untrustworthiness makes a robot more prone to disuse [58] and less likely to be used in cooperative tasks in the future [60]. Explanations can be important [2] for a number of reasons, such as signaling desires and intentions [179], keeping human operators informed and involved in the decision-making loop [322], and reconciling differences between humans' and robots' mental models of the task [65]. When risky situations do arise in safety-critical domains, we expect the robot to identify the issue [323, 324] and explain its reasoning on multiple levels of representation [325, 326]—such as high-level causal symbols and low-level controls [327, 328, 329]—to provide clear explanations for users with different levels of expertise.

Throughout our work, we aim to demonstrate the value of trustable technology and explainable operations in order to promote understanding on human-robot teams (as described in the CURED Framework in Section 1.1.2) and enable robots to earn well-calibrated trust from their human operators.

2.5 Red Teaming

As discussed in Section 2.4.3, every agent in a cooperative task uses models to simplify the unboundedly complex world. While simplifying models are necessary, the incomplete knowledge of these models carries risks, and disastrous outcomes occur when some "unknown unknown" arises that is unaccounted for in the model [60]. In safety-critical domains, we want to minimize the risks inherent in the robot's incomplete knowledge of the world to avoid unsafe situations and dangerous consequences. In Chapter 5, we propose that *safety-aware reasoning* requires *human-robot red teaming* to share autonomy between human and robot agents and challenge the assumptions made by both humans and robots in safety-critical domains. To explore the importance of red teaming within the CURED Framework in this dissertation, we review literature on definitions of red teaming in different fields (Section 2.5.1), different types of red teams (Section 2.5.2), applications of computational red teams (Section 2.5.3), and exploring alternate possibilities through counter-factual reasoning (Section 2.5.4).

2.5.1 Definition of Red Teaming

Red teaming is used to detect weaknesses and vulnerabilities, explore possibilities, consider multiple perspectives or alternate analyses, reveal biases, and challenge conventional wisdom by considering an adversary's perspective [77, 78, 79, 80, 81, 82, 83, 84]. During red teaming exercises, the Blue Team ("good guys") has an objective and considers how the Red Team ("bad guys") may thwart that objective. The Blue Team can improve their plan or capabilities accordingly to prevent the Red Team's attacks [85].

In many red teaming scenarios, especially modeling warfare computationally, the goal is not victory, since winning and losing are not clearly defined. Instead, the ultimate goal of red teaming is to explore possibilities, understand tradeoffs, mitigate risks, educate decision makers, and inform decision-making before disastrous outcomes occur [85, 78]. Red teaming has been applied in a number of domains, including military [78, 85], computer security and cyber-security [86, 77, 80, 87, 82, 88], and even organizational practices and procedures that aim to challenge institutional biases [83].

2.5.2 Types of Red Teams

Red teaming implementations vary greatly depending on the context, but have primarily focused on *human* red teams, where people simulate opponent's viewpoints to challenge their thinking [86, 77, 78, 87, 83, 88]. For example, one work uses a human red team to generate adversarial examples and test the capabilities of a computational language model [84]. More recent work explores *automated* or *computational* red teams to automate the creation of adversarial perspectives in different ways [85, 79]. For example, Perez *et al.* [90] use a red language model to identify offensive language in a target language model. These works aim to use computational red teams to inform the decision-making of human teams [81, 89]. Computational red teams will be discussed in more detail in Section 2.5.3.

Red teaming in complex systems is often less about making decisions and more about exploring the vast space of possibilities [85, 79]. We take inspiration from the idea that red teams serve as "reality checks" and must be involved throughout all stages of a procedure [86]. Previous works on red teaming focus on human red teams, computational red teams, or human teams informed by computational red teams. However, previous research on robot operation in safety-critical problem domains such as space exploration emphasizes the importance of *human-robot* teams [26]. For our work in *safety-aware reasoning* presented in Chapter 5, we propose a *human-robot red team* in which humans and robots work together to provide checks-and-balances and challenge assumptions in both the humans' and robots' simplifying models for shared autonomy tasks.

2.5.3 Computational Red Teams

Human red teams have long been used in military [78] and computer and cyber-security domains [86, 77, 87, 88], and many researchers have explored organizational best practices for allowing red teams to succeed [77, 83]. In contrast, computational red teaming (CRT) has been formalized as *multi-agent systems* comprised of environments, objects, agents, relations, operators, and operations [79].

Abbass *et al.* [79] present multiple levels on which computational red teams may function. Their work emphasizes the importance of defining and modeling computational red teaming problems as multi-agent systems that aim to explore the impact of each agent's actions on the system. The five different levels of computational red teaming represent the increasing degrees to which the computational red team adapts to the complexity of the environment and system being evaluated. The computational red team (CRT) levels are:

a) **CRT0**: An agent is equipped with generic decision-making models—such as reactive rule-based models or finite state machines—that do not evolve as the agent interacts with the environment. Agents at level CRT0 solve relatively simple problems such as fitness or sensitivity analysis.

b) **CRT1**: Each individual agent can learn, adapt, and change its decision-making process as it interacts with the environment. Agents at level CRT1 learn through evolutionary or social/lifelong learning algorithms.

c) **CRT2**: A team of agents learns and evolves together. Teams at level CRT2 explore best configurations or strategies at the team level in order to defend against the fixed strategy of the opposing team.

d) **CRT3**: The environment itself can change and evolve, as opposed to lower CRT levels where agents make decisions within a static environment. Agents at level CRT3 co-evolve alongside the dynamic environment, and the team's performance depends on reciprocal interactions with the other team and the environment.

e) **CRT4**: Agents and teams must reflect in order to identify their own biases and practice "unlearning to learn" [79]. Agents and teams at level CRT4 achieve reflection by self-assessing measures of performance and understanding all components of red teaming, including context, computations, and analysis.

This framework of CRT levels serves to scope how works contribute to red teaming research by describing what levels are relevant to different tasks. In Chapter 5, we will explore different levels of reasoning through which our proposed *human-robot red team* adapts its decision-making.

Computational red teams have been used to solve a wide variety of problems in many applications. Yang *et al.* [85] operate on level CRT2 [79] and use evolutionary algorithms in simulation to learn what blue team battlefield strategies minimize blue team harm against different fixed red team characteristics. The purpose of these computational red teams is to inform and focus human decision-making in different domains [85, 79], for example about physical security assessment of buildings [81] or defending vulnerabilities in large enterprise networks [89].

2.5.4 Counter-Factual Reasoning

Another option for addressing vulnerabilities in a simplified model is to avoid dire consequences by identifying "upstream decision points" [60] through *counter-factual reasoning*. By using near miss events to trigger analysis of the situation that led to dangerous circumstances, counter-factual reasoning can consider past opportunities for action that could be used to avoid future hazards. Counter-factual reasoning is a form of causal inference [330, 331, 332] that learns from experience by considering alternative versions of past events. Counter-factual events are mental representations of alternatives (what might have been), and serve to infer the effects of contrasting actions [333]. In this way, counter-factuals create "blueprints for future action" [334] and explore varying or alternate ways to reach unrealized goals [335, 336].

Due to the importance of explainable robot reasoning [179, 2, 65, 321], some researchers have explored using counter-factual explanations for automated decisions in complex systems. Wachter *et al.* [66] implement human interpretable explanations by modifying inputs to a black-box model, observing the differences in output, and using these differences as counter-factual explanations for the impact of the changes. They note that explanations are crucial for building trust and promoting "societal acceptance of algorithmic decision-making" [66]. In Chapter 5, we explore how *safety-aware reasoning* can utilize counter-factual reasoning by considering alternate risk mitigating actions, thereby identifying upstream decision points to avoid unsafe operating conditions.

Chapter 3

Composable Causality in Semantic Robot Programming

Assembly tasks are challenging for robot manipulation because the robot must reason over the composed effects of actions and execute multi-objective behaviors. Robot control programs are typically written with predefined priorities provided by users to determine how to compose controller behaviors. However, we want the robot to autonomously select these compositions based on their composed effects within the task. We present Composable Causality in Semantic Robot Programming to allow robots to reason over the composed effects of controllers and autonomously compose controllers without predefined priorities. Our proposed *causal control basis* combines controller behaviors with causal information about how the behaviors can be used to execute high-level symbolic actions. The robot uses the *causal control basis* to predict the state-action utility of performing an action to achieve the intended composed effects in the context of the planned assembly task. The composed causality predictions are used to select which action to execute during furniture assembly. We evaluate the robot's state-action utility estimates in different furniture assembly trials in simulation on the Baxter robot. The robot's ability to assemble furniture using different multi-objective connection actions demonstrates the usefulness of the composed causality predictions from our *causal control basis*.

3.1 Introduction

Assembly tasks present unique challenges in reasoning over objects and executing complex behaviors in long-horizon tasks. Robot control programs must overcome difficulties when assembling objects, including composing the effects of multiple behaviors and maintaining these composed effects as they move on to the next step. We need a flexible way to program our robots to perform assembly tasks and Semantic Robot Programming (SRP) [101, 100] has emerged as an intuitive way to declaratively program robots. Within the SRP paradigm, robots can infer goal conditions from a demonstrated goal scene and reason over available objects and actions to reach the goal. The focus of SRP was to address the perceptual challenges involved in perceiving the demonstrated goal conditions. Now that these perceptual challenges have been addressed, we aim to extend SRP in terms of the types of actions that the robot can perform. By extending SRP, we will be able to declaratively program robots to perform actions with composed effects on objects, as is expected in challenging assembly tasks. Specifically, robots need to reason about the objects themselves, compose effects on these objects during task execution, and overcome the challenges of assembly tasks.

Completing an assembly task often requires complex actions, such as achieving concurrent motion goals. For example, screwing in a screw involves moving and performing a spiraling motion concurrently. To perform behaviors while reasoning from the perspective of the object, many works use *object-centric controllers* within a *control basis*. Controllers can be executed concurrently together to execute more complex *composed* behaviors. Compositions of controllers have *multi-objective* roles in achieving the complex motion goals in assembly tasks. We note that the attribute "multi-objective" describes the role of an action in a particular plan. While actions often achieve multiple effects, some of these effects may be side-effects that do not

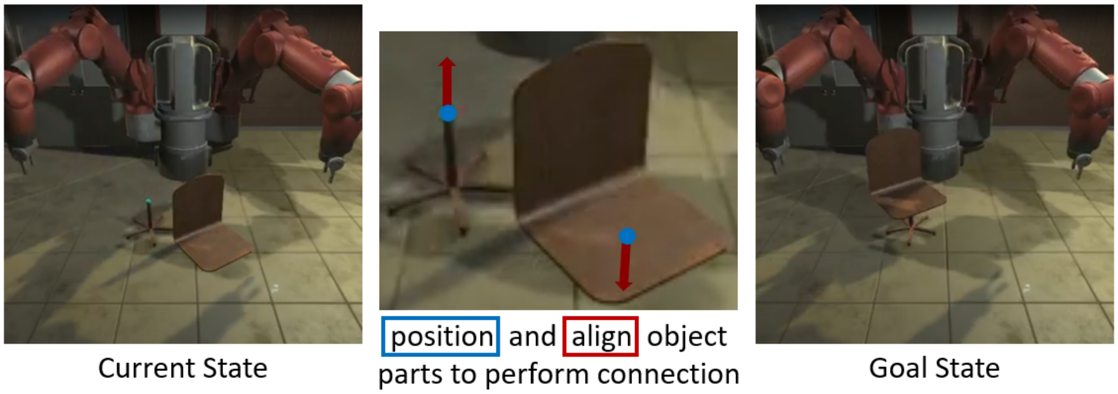


Figure 3.1: A robot assembling furniture through goal-directed manipulation. The robot must reason over the composable causality of controllers to achieve the task goal.

contribute directly toward achieving the task goals while others are relevant to the task objectives. When an action achieves multiple consequences on a causal path toward achieving the plan's goals, we say this action plays a multi-objective role in the plan. Because controllers within a control basis can be composed, object-centric controllers will be the building blocks for actions with multi-objective roles towards achieving assembly task goals.

The compositions of controller behaviors necessary to perform multi-objective actions are generally determined by a predefined priority provided by the user. However, we want robots to autonomously compose controllers without these predefined priorities. Reasoning over actions symbolically can be disconnected from the realities of physical execution. Instead, we expect robots to reason over the pre-conditions and post-conditions of the executable action and ground the action effects on the objects in the perceived scene. Only recently have researchers started to address the question of autonomously composing controllers for multi-objective actions [6]. Figure 3.1 shows an example of the challenges a robot faces while executing a multi-objective action during an assembly task. It is difficult to express the qualitative insights expert users have on controller compositions in a precise, quantitative way that can be reasoned over and used by robots. Instead, we want to place the responsibility on the robot to discover why one controller composition performs better in some situations than others, lessening the cognitive load on the user. The robot will need to reason over the composed effects of different compositions autonomously in order to execute multi-objective actions within the SRP paradigm.

In this chapter, we propose a ***causal control basis*** to build on SRP and allow robots to autonomously compose controllers to achieve assembly tasks. We take inspiration from works that apply notions of *causality* to goal-directed manipulation tasks [8]. The *causal control basis* annotates (possibly composed) controller behaviors with causal information about the roles the controllers play toward achieving task goals. This causal information is intended to allow the robot to determine how to prioritize composed behaviors without relying on predefined priorities from the user. Using the given *causal control basis*, the robot can estimate the utility of a controller composition based on whether the composition is predicted to achieve the desired composed effects on the objects in the scene. During task execution, the robot will autonomously compose controller behaviors based on their predicted composed effects and execute the afforded multi-objective actions to assemble a piece of furniture. We test the causal reasoning in simulation on the Baxter robot in a variety of furniture assembly tasks and find that it provides the robot with sufficient information to autonomously compose controllers without predefined priorities from the user. Our work on ***composable causality*** demonstrates that the *causal control basis* allows the robot to achieve challenging goal-directed manipulation tasks within the SRP paradigm.

3.2 Related Work

3.2.1 Assembly Tasks

We consider construction tasks as an interesting domain for complex goal-directed manipulation. Assembly is a challenging problem because it requires the ability to plan over long-horizons, understand properties of objects, and manipulate objects in particular ways. Nair *et al.* [13] present a tool construction pipeline that allows robots to construct tools to achieve tasks. Many works use learning from demonstration to teach reusable motion primitives that the robot uses to assemble different objects [173, 172], including furniture such as tables [171]. Lee *et al.* [9, 10] developed the IKEA Furniture Assembly Environment as a test-bed for the perception, planning, and control required to perform construction tasks. Though their assembly environment is designed for reinforcement learning, we use the IKEA Furniture Assembly Environment to allow the robot to predict the effects of a given control policy. Together, these works emphasize the immense interest and challenge of tool-use and construction tasks, which motivates our choice of assembly tasks as our problem domain.

3.2.2 Object-Centric Controllers and Control Basis

Due to the significance of object affordances [33], robotics research has gravitated towards interacting with objects through *object-centric* motions. Object-centric behaviors are expressed in *task frames* that bridge the gap between high-level symbolic description of actions and the low-level servomechanism execution of actions [5, 122, 123]. For example, expressing a crank-turning-action in world frame involves reasoning over the arc the crank might follow. But expressing the same crank-turning-action in a task frame fixed to the crank involves applying force along an axis of the crank until resistance is met [5]. Reasoning in the task frame simplifies the expression of the action and emphasizes the effect on the object itself.

To execute these object-centric actions, researchers use object-centric controllers, which send joint commands such that the robot achieves low-level motion primitives. Controllers can be used within a *control basis* that forms the building blocks of all behaviors the robot might need to execute. The behaviors within a control basis can be *composed* through *nullspace projection* to yield multi-objective behaviors, enabling

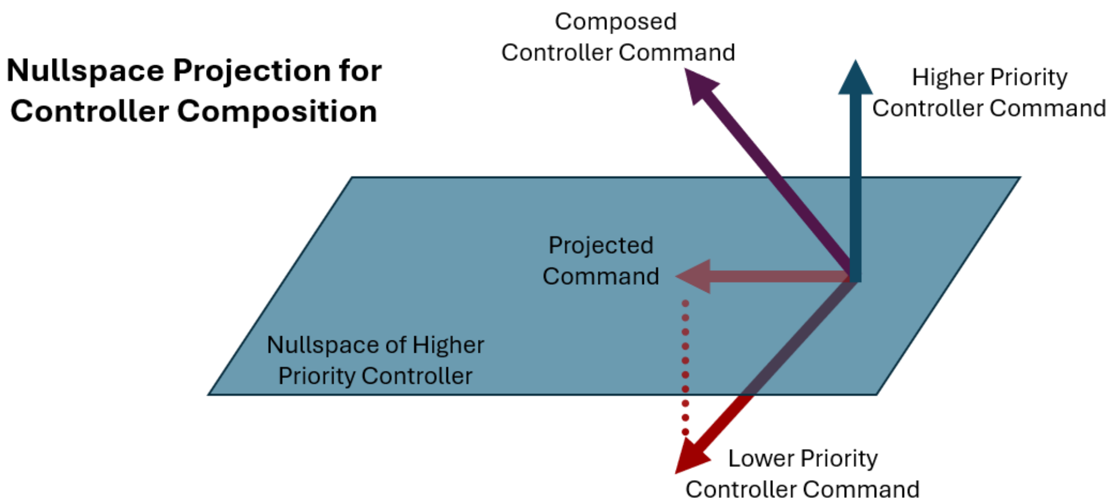


Figure 3.2: Visualization of controller composition, where two controller commands are prioritized so they may be executed concurrently. The prioritization allows the commands to be combined through nullspace projection. The visualization shows the resulting composed controller command.

multiple controllers to be executed concurrently [186] to achieve multiple composed goals [7, 41]. For example, robust grasping could be formulated as a multi-objective action that involves *positioning* an end-effector while *aligning* the approach axis of that end-effector with the target object. Composing multiple behaviors induces a priority between these behaviors so that the controllers do not prevent each other from converging. Mathematically, a priority between controllers means the joint commands from the lower priority controller will be projected into the nullspace of the higher priority controller. Section 3.3.2.1 will describe the prioritization and projection in more detail. Practically, a priority between controllers means that any progress made towards achieving the lower priority controller objective will not disturb or disrupt any progress made towards the higher priority controller objective. Figure 3.2 presents a visualization of how prioritized controllers can be composed using nullspace projection to result in a composed controller command. The priorities between controllers—the particular composition of these controllers—can greatly impact the multi-objective role that action plays in the task plan. Many works explore composing controllers in atomic actions such as grasping [46, 45, 47] or conditioning behaviors [48] such as avoiding joint limits and singularities.

Many works compose controller behaviors using predefined priorities provided by the user [48, 46, 45, 47]. For example, a user may determine experimentally that prioritizing positioning over alignment—where prioritizing the controllers through nullspace projection is not a sequential relation, and instead the controllers execute concurrently in a way that does not disrupt each other—results in the most robust grasp poses. Therefore, the user will hard-code the robot to always perform multi-objective grasps by composing these behaviors such that positioning is the highest priority. However, we want robots to autonomously compose controllers to increase their reasoning power over complex manipulation actions. Sharma *et al.* [6] present a reinforcement learning approach to determining how controllers should be composed to perform different tasks. They demonstrate that their approach allows robots to autonomously compose controllers in atomic actions such as block pushing, screw turning, and door opening. The work of Sharma *et al.* [6] demonstrates a significant step towards autonomously composing controllers in order to perform atomic actions. We build on this work and extend it to long-horizon construction tasks, where it is necessary to reason about how the multi-objective action will be used in sequence to achieve a larger task goal. This distinction requires the robot to have a deeper understanding of when controller behaviors can be enacted, what the composed effects of multi-objective actions will be, and how to maintain these composed effects throughout task execution.

3.2.3 Causality

Work on causality analyzes cause and effect relationships between variables. Causal relationships can be expressed as Causal Bayesian Networks (CBNs) [337] and analyzed through queries [330, 331, 332]. Traditionally, work on *causality* refers to the

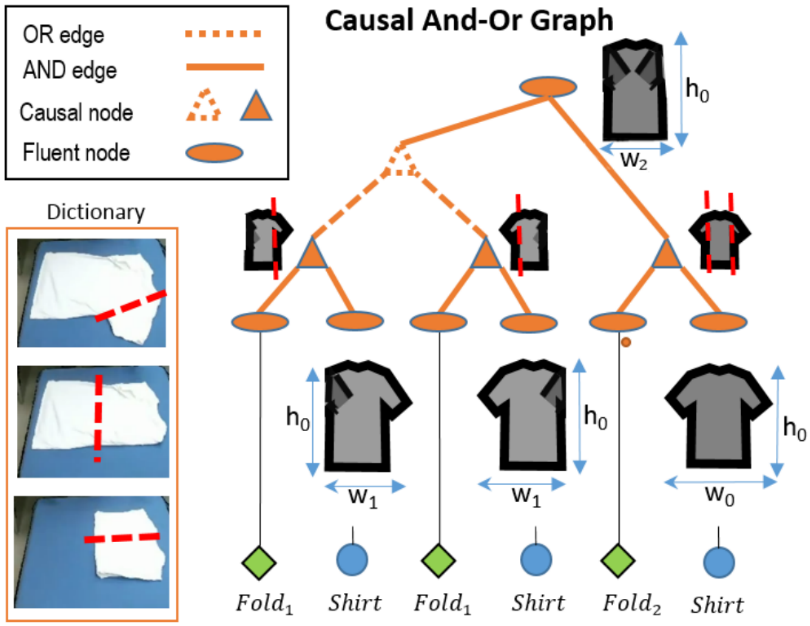


Figure 3.3: A Causal And-Or Graph presented by Xiong *et al.* [8]. The causal graph describes how actions (causal nodes) induce changes in the state (fluent nodes). The pictured causal graph can be used in laundry folding tasks. For example, to fold both sleeves on a shirt (root fluent node), the robot must have folded each sleeve in individually (as indicated by the causal OR node in the left subtree) or folded both sleeves together (as indicated by the causal AND node in the right subtree). A key feature of the causal graph is to indicate multiple potential ways to achieve the same root effect.

effects of changes to variables on the distributions of other random variables in the system. In the context of robot manipulation, *causality* takes on a different meaning, referring to how robot actions cause effects on objects in the scene. Figure 3.3 shows an example causal graph from Xiong *et al.* [8] that reasons over desired effects, possible actions to induce those effects, and the starting state from which those actions could be taken. Causality has been particularly helpful in allowing robots to reason over long-horizon tasks, such as assembly tasks.

Work in robot manipulation often uses several causal models. Xiong *et al.* [8] found that hierarchical spatial, temporal, and causal models can be learned from demonstration and used to achieve cloth-folding tasks. We build on this work by incorporating hierarchical temporal information and causal models into our proposed *causal control basis* to allow robots to autonomously compose controllers and reason over the composed effects of actions in long-horizon tasks. In particular, the causal relationships captured in the *causal control basis* will be similar to those in Figure 3.3 and capture action pre-conditions, possible actions to be taken, and the induced effects of those actions.

3.3 Methods

3.3.1 Problem Formulation

To perform assembly tasks that require multi-objective behaviors, the robot needs to predict the *state-action utility* of executing each controller composition in the current state based on whether the given composition of controllers is predicted to achieve its composed effects. By estimating the state-action utility, the robot will evaluate how valuable it is to execute a particular controller composition; if the composition achieves its intended effects and makes progress towards achieving the task goals, then this action has a high state-action utility from the current state.

We assume we have a control basis Φ of controllers that can be composed to achieve multiple objectives (the methodology for composition will be described in Section 3.3.2.1). Given a task goal, the robot constructs a high-level task plan using an off-the-shelf task planner and decomposes each symbolic action into a sequence of executable motions. For actions that require multi-objective behaviors, we want the robot to autonomously compose the given controllers and execute the planned symbolic action by reasoning over the *causality* of the controllers based on the utility estimates.

We formulate this probabilistic planning problem as a Markov Decision Process (MDP) [338, 184, 339, 340] (S, A, P, R). The continuous state of the world is defined by the poses of the objects $\mathbf{x}_{\text{obj}} \in SE(3)$ and the robot configuration $\mathbf{q}_t \in C$ for configuration space C. A state s in state space S is a discrete, symbolic representation of the continuous state of the world. A control basis is a set of controllers $\Phi = \{\phi_i\}$. An action a in action space A is the execution of a (possibly composed) controller behavior. For controller behavior $\phi_i \in \Phi$, we denote an action $a_i = \texttt{execute}(\phi_i)$. The controllers that can be running at any given time are elements of the power set of the control basis $\mathcal{P}(\Phi)$. Suppose we have controllers ϕ_i and ϕ_j that achieve objectives i and j, respectively. One possible composition of these controllers is $\phi_j \lhd \phi_i$, where the "subject-to" relation $\lhd$ indicates that ϕ_i has a higher priority than ϕ_j (or equivalently, that ϕ_i is higher priority than ϕ_j). Let $B_t \in \mathcal{P}(\Phi)$ be the set of controllers running at time t. Since composing controllers induces an ordering (priority) between them, all possible compositions of the running controllers B_t are elements of the symmetric group $\mathcal{S}_{B_t}$, which is the set of all possible permutations over the elements (controllers) in B_t. For example, if $B_t = \{\phi_i, \phi_j\}$, then $\mathcal{S}_{B_t} = \{\phi_i \lhd \phi_j, \phi_j \lhd \phi_i\}$. Therefore, the set of all possible composed controller behaviors Ξ for a control basis Φ is:

$$\Xi = \{\mathcal{S}_{B_t} \;\mid\; B_t \in \mathcal{P}(\Phi)\} \tag{3.1}$$

The resulting action space is:

$$A = \{\texttt{execute}(\xi) \;\mid\; \xi \in \Xi\} \tag{3.2}$$

The transition probability $P(s'|s, a)$ indicates the probability of achieving the composed symbolic effects s' of a (composed) controller execution $a \in A$ when enacted in the current state s. The transition probability will be discussed more in Section 3.3.4. The reward function $R(s, a, s')$ is the reward of performing action $a = \texttt{execute}(\phi_i)$ that executes controller ϕ in state s to achieve composed symbolic effects s'. We define the reward function:

$$R(s, a, s') = \begin{cases} 1 & \text{controller objectives met} \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where the reward is 1 if the action achieves its intended effects s' (the executed controllers achieve their objectives) and the reward is 0 otherwise. We want the robot to execute the controller ϕ_i that has the highest state-action utility (cumulative reward) based on whether the controller is predicted to achieve its composed effects.

The action space is determined by the *causal control basis*, but the robot is not given any information about the state-action utilities associated with the (composed) controllers. The robot needs to estimate the state-action utilities for each possible controller composition.

3.3.2 Causal Control Basis

We propose a *causal control basis* $\mathbf{\Phi}$ that the robot will use to predict the state-action utilities of actions and determine which composition of controllers to execute to achieve assembly tasks. The *causal control basis* $\mathbf{\Phi} = (\Phi, \mathcal{T}, \mathcal{G}_\mathcal{C})$ is given to the robot and is comprised of the following components: a set of controllers or control basis $\Phi = \{\phi_i\}$, a set of temporal action sequences $\mathcal{T} = \{\tau\}$, and a set of composed causal graphs $\mathcal{G}_\mathcal{C} = \{G_C\}$. The following sections (Section 3.3.2.1, Section 3.3.2.2, and Section 3.3.2.3) describe each component of the *causal control basis* (Φ, $\mathcal{T}$, and $\mathcal{G}_\mathcal{C}$, respectively) in greater detail.

3.3.2.1 Control Basis

The control basis Φ is a set of controllers $\Phi = \{\phi_i\}$ that form the building blocks of the robot's behaviors. The control *basis* is related to basis vectors in a vector space. A basis in a vector space is a set of vectors from which any vector in the vector space can be written as a linear combination of the vectors in the basis. Similarly, as described in Section 3.2.2, controllers in a control basis can be *composed* or *executed concurrently* to perform more complex behaviors that achieve composed effects. Joint commands from multiple controllers can be combined using several different schemes. For example, joint commands could be added together. However, conflicting joint commands can prevent either controller from converging and achieving the controller objective. Instead, several works use *nullspace projection* (or equivalently *nullspace composition*) to compose commands from multiple controllers [46, 45, 47].

Nullspace composition induces a priority between the concurrently executed controller behaviors. The "subject-to" relation $\triangleleft$ describes this prioritized relationship. For example, for two controllers $\phi_i, \phi_j \in \Phi$, the relation $\phi_j \triangleleft \phi_i$ indicates that the objective of controller ϕ_j will be performed "subject-to" the objective of controller ϕ_i. Equivalently, we say that controller ϕ_i is higher priority that ϕ_j. Figure 3.2 provides a visualization of how nullspace composition creates composed controller commands.

Computationally, nullspace composition projects lower priority controller commands into the nullspace of higher priority controller commands. This nullspace projection ensures that the lower priority controller commands are tangential to the higher priority commands. Consider the controller composition $\phi_j \triangleleft \phi_i$. The controllers ϕ_i and ϕ_j compute joint commands $\Delta\mathbf{q}_i$ and $\Delta\mathbf{q}_j$, respectively. Nullspace composition will project the lower priority controller command $\Delta\mathbf{q}_j$ into the nullspace $\mathcal{N}$ of the higher priority controller ϕ_i. We can compute the Jacobian $\mathbf{J}_i$ of the control function ϕ_i:

$$\mathbf{J}_i = \frac{\partial \phi_i}{\partial \mathbf{q}} \tag{3.4}$$

which denotes how the controller command changes with respect to the robot configuration $\mathbf{q}$. The controller Jacobian can be used to compute the nullspace of

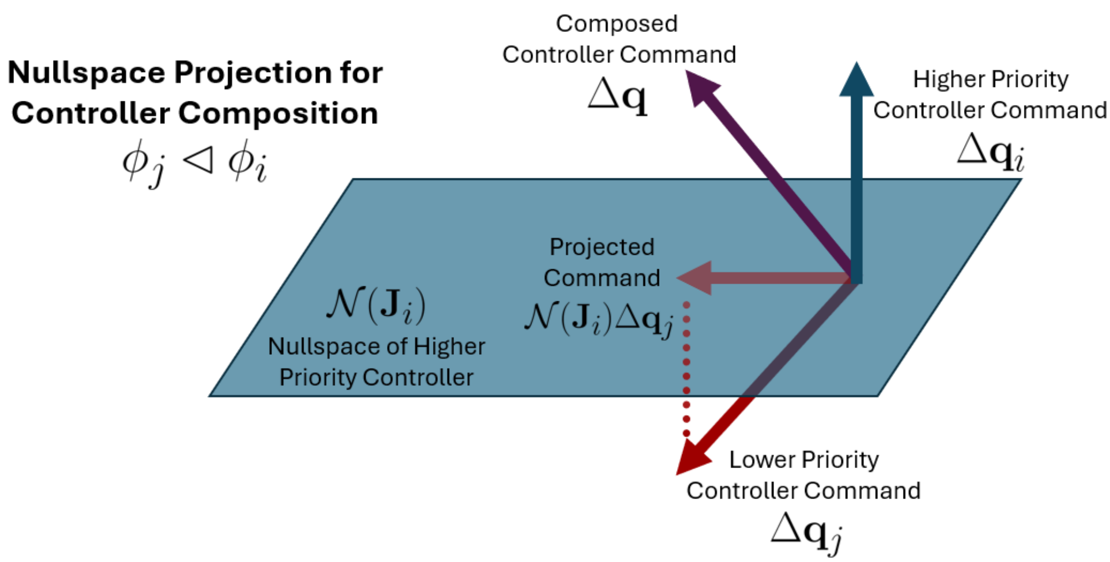


Figure 3.4: Visualization of controller composition $\phi_j \lhd \phi_i$ through nullspace projection and the resulting composed controller command $\Delta\mathbf{q}$.

the higher priority controller:

$$\mathcal{N}(\mathbf{J}_i) = \mathbf{I} - \mathbf{J}_i^{\#}\mathbf{J}_i \tag{3.5}$$

where $\mathbf{I}$ is the identity matrix and $(\cdot)^{\#}$ denotes the matrix pseudo-inverse. The combined joint command $\Delta\mathbf{q}$ can be computed using the nullspace projection:

$$\Delta\mathbf{q} = \Delta\mathbf{q}_i + \mathcal{N}(\mathbf{J}_i)\Delta\mathbf{q}_j \tag{3.6}$$

Figure 3.4 shows a visual representation of the nullspace projection for controller composition $\phi_j \lhd \phi_i$. Nullspace projections can be chained together to compute composed commands for more complex controller compositions. For example, for controller composition $\phi_k \lhd \phi_j \lhd \phi_i$, the combined joint command $\Delta\mathbf{q}$ can be computed using multiple nullspace projections:

$$\Delta\mathbf{q} = \Delta\mathbf{q}_i + \mathcal{N}(\mathbf{J}_i)\left[\Delta\mathbf{q}_j + \mathcal{N}(\mathbf{J}_j)\Delta\mathbf{q}_k\right] \tag{3.7}$$

Now that we have presented the computational methodology behind nullspace composition, we aim to provide more intuitive understanding of the effect of nullspace composition on the executed robot behavior. Composition of controller behaviors means the controllers are executed concurrently. The nullspace projection method for controller composition ensures that the combined control commands do not undo progress made toward achieving each controller objective. In particular, the prioritization of controllers $\phi_j \lhd \phi_i$ means that any progress made toward the lower priority controller objective ϕ_j does not undo any progress made toward the higher priority controller objective ϕ_i. Nullspace projection projects the lower priority controller command into the *nullspace* of the higher priority controller command, which results in a projected controller command that is orthogonal to the higher priority controller command. We can think of nullspace projection as affecting the time-scales on which the controller commands converge. In general, the nullspace projection of the lower priority controller means that the lower priority controller will approach its convergence point at a slower time-scale. The projection prevents the robot from executing the full joint command, and slows the robot's progress towards that controller goal. The higher priority controller will approach its convergence point at a faster time-scale, since the commands are not projected.

Several works use the nullspace composition approach for combining controller commands such that the resulting robot behaviors achieve multi-objective roles within the task plan [45, 46, 47]. The goal of our work is to provide robots with the reasoning capabilities necessary to determine the appropriate prioritization of controller behaviors that will achieve the desired composed effects.

For the tasks considered in the experiments, we define the control basis—discussed in more detail in Section 3.4.1.1—to include controllers for gripper joints, end-effector pose, end-effector position, end-effector rotation, and screwing behaviors. However, our formulation will work with an arbitrary control basis.

3.3.2.2 Temporal Action Sequences

The set $\mathcal{T}$ of temporal action sequences $\mathcal{T} = \{\tau\}$ is the set of the sequences of controller executions that correspond to high-level symbolic actions. This temporal information can be presented in several ways, such as behavior trees or hierarchical temporal graphs [8]. For simplicity, we use functions τ to encode the list of actions the robot should execute to achieve specific symbolic effects. Every temporal action sequence function τ includes the high-level symbolic action name, the intended action effect, and, most importantly, the list of low-level actions the robot must execute. The low-level actions in the temporal action sequence all have the form $a = \texttt{execute}(\phi)$, as described in Section 3.3.1. The controllers to be executed can be primitives (such as opening and closing a gripper), simple controller behaviors (such as moving the robot's end-effector to a given 6D pose), or *composed* controller behaviors.

The purpose of the set of temporal action sequence functions $\mathcal{T}$ is to describe how the controllers from the control basis Φ can be sequenced to perform high-level symbolic actions. These temporal action sequence functions will be used to inform the robot's action sequences to complete task goals.

Figure 3.5 presents a general form for the pseudocode of the temporal action sequence functions τ. Figure 3.6 shows an example temporal action sequence function $\tau_{\text{pick-up}}$, which provides the sequence of low-level actions to execute in order to achieve the high-level symbolic pick-up action.

3.3.2.3 Composed Causal Graphs

The set $\mathcal{G_C}$ of composed causal graphs $\mathcal{G_C} = \{G_C\}$ is the set of composed causal graphs that describe the desired effects of controller compositions with multi-objective roles in the task plans. A composed causal graph $G_C \in \mathcal{G_C}$ is composed of 3 types of nodes (Figure 3.7a):

```
action-sequence-function high-level-action-name(input-parameters):
    effects: [
        // list of symbolic effects
    ]
    return [
        // list of actions of the form:
        // execute(controller(parameters))
    ]
```

Figure 3.5: The general form of a temporal action sequence function τ. Each programmatic function has a name representing the high-level action, function inputs, and intended symbolic effects. The function τ returns the sequence of low-level actions required to perform the high-level symbolic action. Each low-level action is an execution of a controller behavior from the control basis Φ.

```
action-sequence-function pick-up(object=obj, end-effector=ee):
    effects: [in-hand(ee, obj)]
    return [
        execute(φ_gripper(open(ee))),
        execute(φ_6Dpose(pre-grasp(obj))),
        execute(φ_6Dpose(grasp(obj))),
        execute(φ_gripper(close(ee))),
        execute(φ_6Dpose(post-grasp(obj)))
    ]
```

Figure 3.6: The temporal action sequence function $\tau_{\text{pick-up}}$ for the high-level `pick-up` action. This action sequence picks up an object `obj` with robot end-effector `ee` by executing controllers $\phi_{\text{gripper}}, \phi_{\text{6Dpose}} \in \Phi$, which send joint commands to the gripper and move the end-effector to a 6D pose, respectively. Functions `open` and `close` provide the joint states for the given end-effector corresponding to the primitives open and close. We assume we have functions that determine the `pre-grasp`, `grasp`, and `post-grasp` poses relative to a given object `obj`.

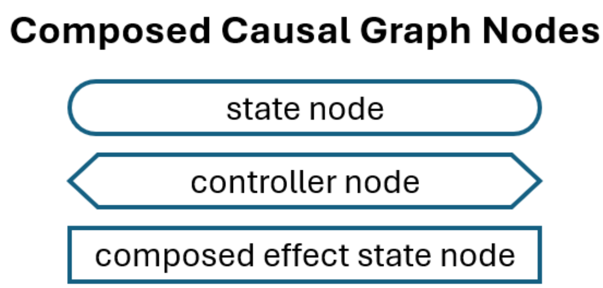
- **State nodes** that are literals describing a symbolic state;
- **Controller nodes** that indicate a controller that will cause a change of the symbolic state; and
- **Composed effect state nodes** that describe the desired composed effects of an action with a multi-objective role in the plan.

The nodes are organized in levels, meant to indicate temporal relationships. State and/or composed effect state nodes on the same level of the graph indicate that these literals are all true at the same time. Controller nodes on the same level of the graph indicate that these controller behaviors could be execute concurrently or composed to achieve a desired composed effect. The nodes in the composed causal graph are connected by 3 types of directed edges (Figure 3.7b):

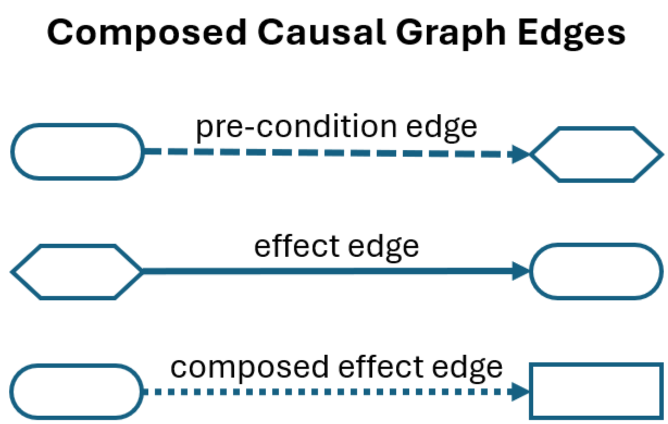
- **Pre-condition edges** that connect state nodes to controller nodes, indicating that the state is a pre-condition for executing the controller;
- **Effect edges** that connect controller nodes to state nodes, indicating that the state is a post-condition or consequence of the executed controller; and
- **Composed effect edges** that connect state nodes to composed effect state nodes, indicating that some combination of states could result in the desired composed effects.

Figure 3.7 shows a legend of the nodes and edges that will be used to create composed causal graphs.

The purpose of a composed causal graph G_C is to provide information to the robot about the intended multi-objective role a composed controller plays in a task plan. Specifically, the composed causal graph indicates the pre-conditions and desired composed effects of a composed controller behavior. However, the composed causal graphs *do not* indicate how to compose the controllers or what



(a) Types of nodes in a composed causal graph $G_C \in \mathcal{G}_\mathcal{C}$.



(b) Types of edges in a composed causal graph $G_C \in \mathcal{G}_\mathcal{C}$.

Figure 3.7: Legend of types of nodes and edges used to create composed causal graphs.

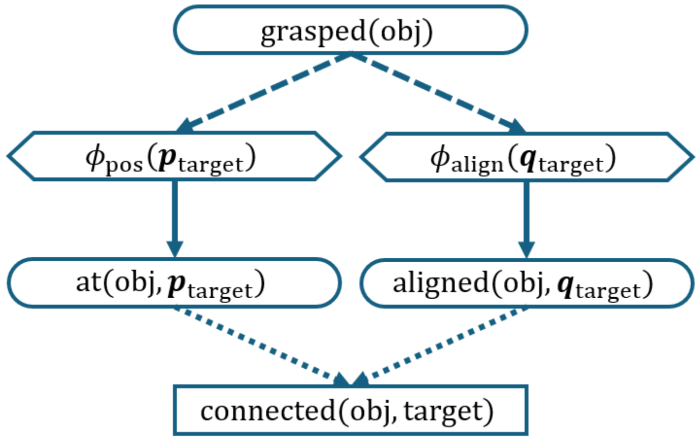


Figure 3.8: Example composed causal graph for a generic connection action that connects an object `obj` to a target piece `target`. Initially, we expect the object `obj` to be grasped by the robot. Grasping is a pre-condition for executing positioning ϕ_{pos} and alignment ϕ_{align} controllers. The controllers will achieve the effects that the object is positioned and aligned relative to the target based on 3D position $\boldsymbol{p}_{\text{target}}$ and quaternion $\boldsymbol{q}_{\text{target}}$. If the object `obj` is appropriately positioned and aligned relative to the target object `target`, then the robot will achieve the composed effect that the two parts are connected. This composed causal graph indicates to the robot that the composed effect `connected(obj, target)` can be achieved by composing the controllers ϕ_{pos} and ϕ_{align}. However, the composed causal graph does not tell the robot how to appropriately prioritize these behaviors. The robot must determine which prioritization—$\phi_{\text{align}} \lhd \phi_{\text{pos}}$ or $\phi_{\text{pos}} \lhd \phi_{\text{align}}$—to execute in order for this composed controller behavior to achieve the appropriate multi-objective role in the assembly task plan.

the prioritization between the controller behaviors should be in order to achieve the desired composed effects. Instead, this causal information will allow the robot to estimate the state-action utility of executing different controller compositions based on which compositions are most likely to achieve the desired composed effects indicated in the graph.

As an example of the relationships presented in a composed causal graph, Figure 3.8 presents a composed causal graph for a generic connection action, which will attempt to connect two pieces of furniture by composing positioning and alignment controllers. Note that the composed causal graph does not indicate how the robot should compose or prioritize these controllers in order to achieve the part connection.

The composed causal graph provides the relevant causal information that will enable the robot to reason autonomously over composing the controllers without pre-defined priorities from the user.

3.3.3 State-Action Utility Predictions

Given the domain-specific *causal control basis* $\mathbf{\Phi}$, the robot must estimate the utility of executing the possible controller compositions described by the set $\mathcal{G}_{\mathcal{C}}$ of composed causal graphs. In particular, whenever two controller nodes exist on the same level of a composed causal graph $G_C \in \mathcal{G}_{\mathcal{C}}$—indicating that the controllers are executed concurrently—the robot will consider all possible compositions of those behaviors. The robot will predict the state-action utility of executing a specific prioritization of the controllers in the current state to achieve the expected composed effects indicated in G_C.

To predict the state-action utilities of the possible controller compositions described by the set $\mathcal{G}_{\mathcal{C}}$ of composed causal graphs in the *causal control basis*, the robot will perform a large number N of Monte Carlo simulations. Suppose the robot is predicting the state-action utility of arbitrary controller composition $\phi_k \lhd \phi_j \lhd \phi_i$ (where the "subject-to" relation $\lhd$ indicates the priority of behaviors in the composition, specifically that ϕ_i is the highest priority approaching its goal at the fastest time-scale, and ϕ_k is the lowest priority approaching its goal at the slowest time-scale) suggested by causal graph G_C. For each Monte Carlo simulation, the robot uniformly samples its initial starting configuration $\mathbf{q}$ and poses of the objects in the scene $\{\mathbf{x}_{\text{obj}}\}$, which will determine the symbolic starting state $s \in S$. The continuous goals for each controller are determined by the expected final object poses, which will also be uniformly sampled such that the expected symbolic composed effects $s' \in S$ can be achieved, where s' is the composed effect node in G_C. During the Monte Carlo simulation, the robot will simulate execution of action

$a = \texttt{execute}(\phi_k \lhd \phi_j \lhd \phi_i)$ until the composed controllers converge or until large time threshold T. The *causal control basis* computes a utility for the simulation based on the composed controller behavior in the simulated scenario. Let $\hat{Q}_n(s,a)$ be the predicted state-action utility [339, 340] for simulation n of N. We define the predicted state-action utility to be:

$$\hat{Q}_n(s,a) = \begin{cases} 1 & \text{controller objectives met} \\ 0 & \text{bad progress} \\ h_a(s,s',s_T) & \text{early termination, } t \geq T \end{cases} \tag{3.8}$$

where s_T is the state at time threshold T when the simulation ends and $h_a(s,s',s_T)$ is a heuristic evaluation function based on starting state s, the target simulation end state s', and the actual simulation end state s_T at time $t = T$. When the controllers meet their objectives, the simulation earns a predicted normalized utility of $\hat{Q}_n(s,a) = 1$ since the simulated action execution achieved the intended composed effects (earning reward $R(s,a,s') = 1$ according to Equation 3.3). If bad progress is made, then at some state s_t where $t < T$, the controllers reached a local minimum—for example, the controller objective function stopped decreasing. Bad progress results in a predicted normalized utility of $\hat{Q}_n(s,a) = 0$, indicating the simulation did not make progress towards the composed goals (earning reward $R(s,a,s') = 0$ according to Equation 3.3). Finally, if the simulated time threshold is exceeded $t \geq T$, then we terminate the simulation early and estimate the utility value with a heuristic evaluation function [339, 340]. We utilize a heuristic that computes the percent decrease of the distance to the goal:

$$h_a(s,s',s_T) = \frac{d_a(s,s') - d_a(s,s_T)}{d_a(s,s')} \tag{3.9}$$

where the distance function d_a computes the controller-specific distance between an initial state and an ending state. This heuristic function h_a is meant to estimate progress towards the goal made by the controller composition in the case where the simulation terminates before convergence.

The *causal control basis* estimates the state-action utility of executing action $a = \texttt{execute}(\phi_k \lhd \phi_j \lhd \phi_i)$ based on initial state s and intended composed effects s'. The estimated utility for N Monte Carlo simulations [339, 340] is:

$$\hat{Q}(s,a) = \frac{1}{N} \sum_{n=1}^{N} \hat{Q}_n(s,a) \tag{3.10}$$

During task execution, whenever the robot has to perform a multi-objective action, the robot will query the *causal control basis* for the action that has the highest state-action utility. Using these predicted state-action utility values, the robot will choose to execute the action a according to a policy π that selects the action with the highest predicted state-action utility:

$$a = \pi(s) = \arg\max_{a} \left(\hat{Q}(s,a)\right) \tag{3.11}$$

toward achieving the intended composed effects.

3.3.4 Action Repetition

As will be described in Section 3.4.4, when an action fails, we allow the robot to retry the action to complete the assembly task goals. In Section 3.3.1, we defined $P(s'|s,a)$ to be the probability that action a achieves the (possibly composed) symbolic effects s' when executed in state s. The probability that an action a fails to achieve its intended effects is $1 - P(s'|s,a)$. Let $P_z(s'|s,a)$ be the probability that action a achieves its intended effects after z attempts. If the action fails $z-1$

times and succeeds on attempt z, the repeated action will succeed with probability:

$$P_z(s'|s,a) = 1 - [1 - P(s'|s,a)]^z \tag{3.12}$$

If we allow enough attempts of the action, this probability will approach 1, such that $P_z(s'|s,a) \to 1$. Ideally, we want actions to succeed after few attempts in order to complete the assembly task more efficiently. However, even if the action occasionally fails, we do expect the robot to eventually complete the assembly task after multiple action attempts according to Equation 3.12.

3.4 Experiments and Results

Figure 3.9 describes the pipeline for assembling furniture using Composable Causality in Semantic Robot Programming and the use of our proposed *causal control basis*. We assume that the robot has parsed the goal conditions from a demonstrated goal scene of the task as in SRP [101, 100]. We also assume that we have affordance-based perception [129, 141, 143] to perceive the objects and affordances in the scene. For example, Affordance Coordinate Frames (ACFs) [140, 141] are one possible implementation for affordance-based perception. These perceived objects and affordances seed the initial state of an off-the-shelf high-level task planner[1] [341], which constructs the task plan. The temporal action sequence functions from the *causal control basis* tell the robot how to convert each high-level action in the task plan into a sequence of (possibly composed) low-level controller commands. The robot instantiates each low-level action based on the current poses of the objects and their perceived connection sites. To execute the connection actions with multi-objective roles in the task plan, the robot queries the *causal control basis* for the action with the highest predicted state-action utility according to Equation 3.11. The robot

[1]Pyperplan STRIPS planning library: https://github.com/aibasel/pyperplan

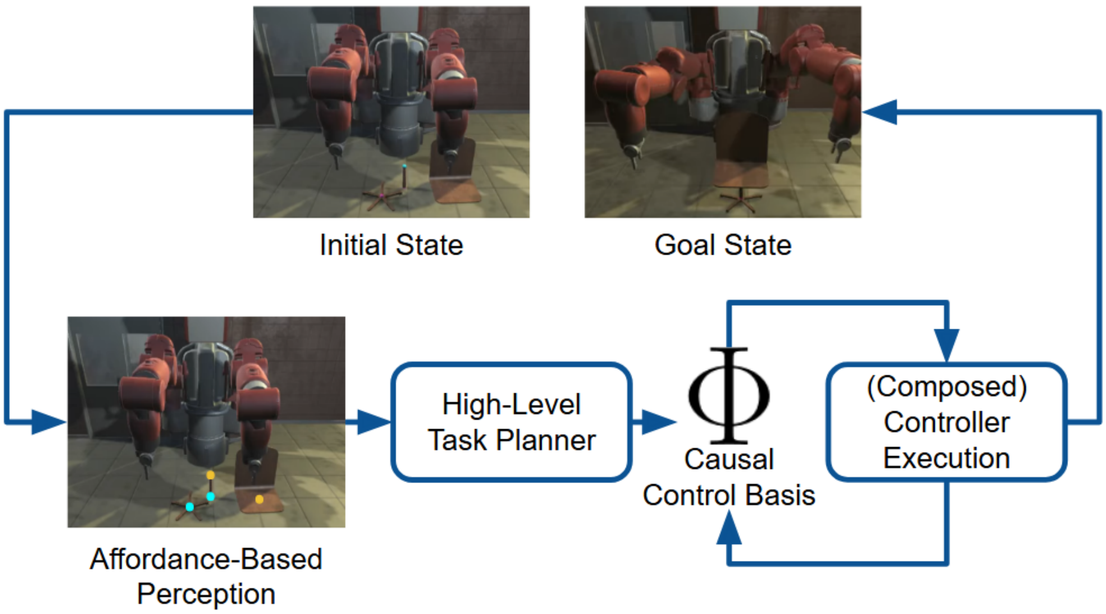


Figure 3.9: Our pipeline for Composable Causality in Semantic Robot Programming. From the initial state, the robot perceives the affordances in the scene. In the depicted example, the robot perceives the afforded connections between the base of the chair and the support rod (in blue) and between the support rod and the chair seat (in yellow). These perceived affordances inform the high-level task planner, which creates a task plan. For example, the task planner may produce the high-level plan `[pick-up(rod, right-hand), connect(rod, base), pick-up(seat, left-hand), connect(seat, rod)]`. The task plan is executed by querying the *causal control basis* for temporal action sequences, which determine the low-level controller behaviors required to execute each high-level action (`pick-up` and `connect`) from the task planner. For actions that play a multi-objective role in the plan by connecting two pieces of furniture, the *causal control basis* is queried again to find the executed controller composition with the highest utility (for example, the *causal control basis* may estimate that in order to connect two parts based on positioning and alignment, the action `execute`($\phi_{\text{align}} \triangleleft \phi_{\text{pos}}$) has the highest predicted state-action utility). Possibly composed controllers are executed according to the temporal and causal information in the *causal control basis* until the goal state is reached.

executes this sequence of controllers to achieve the task goal of assembling furniture.

We evaluate the proposed *causal control basis* in various furniture assembly tasks in simulation using the Baxter robot in the IKEA Furniture Assembly Environment[2] [9, 10]. We assume that known object poses during manipulation and grasp poses for every object part are provided. The connection of two parts is implemented as welding in the Mujoco simulation, which checks the position and axis alignment of connecting points.

Note that the simulated welding simplifies the furniture assembly problem; what the simulator detects as sufficient positioning and alignment of the connecting points would, in most cases, not translate to a sufficient part connection in a real-world assembly task. These complex connection actions are their own area for future research, since maintaining positioning and alignment while performing the connection operation with a limited set of robot end-effectors is a whole-body control problem. We leave real-world part connections for future work. Our experiments using the *causal control basis* are meant to demonstrate the feasibility of autonomously composing the controllers that will help achieve real-world part connections.

3.4.1 Causal Control Basis for Furniture Assembly

3.4.1.1 Control Basis Implementation

In this work, we define the control basis Φ for furniture assembly by gripper joints ϕ_{gripper}, 6D end-effector pose ϕ_{6Dpose}, 3D end-effector position ϕ_{pos}, end-effector rotation ϕ_{rot}, and screw ϕ_{screw} controllers. The gripper joint controller ϕ_{gripper} commands the gripper to configurations such as `open` and `close`. The 6D pose controller ϕ_{6Dpose} commands the robot to reach an end-effector toward a given pose. The 3D position controller ϕ_{pos} is used to position objects close to one another, specifically to achieve the proximity constraint used for connecting two object parts.

[2] https://clvrai.github.io/furniture/

The rotation controller ϕ_{rot} achieves relative orientation constraints between object parts. The screw controller ϕ_{screw} is used specifically for screwing in one object to a target object, such as screwing a table leg into the tabletop. The rotation controller would be used to maintain alignment between the table leg and tabletop while the screw controller performs the screwing motion and applies pressure to achieve the connection.

All of these controllers are implemented as object-centric potential field controllers based on attractive potential fields that attract the robot and objects to the controller goal. Our furniture assembly control basis Φ is the set of these controllers:

$$\Phi = \{\phi_{\text{gripper}}, \phi_{\text{6Dpose}}, \phi_{\text{pos}}, \phi_{\text{rot}}, \phi_{\text{screw}}\} \tag{3.13}$$

3.4.1.2 Temporal Action Sequences

The set $\mathcal{T}$ of temporal action sequence functions $\mathcal{T} = \{\tau\}$ indicate the sequence of controllers that correspond to the high-level `pick-up`, `insert`, and `screw` actions. Figure 3.6 shows the high-level `pick-up` temporal action sequence and Figure 3.10 shows the high-level temporal action sequences for the connection actions `insert` and `screw`. For the connection actions `insert` or `screw`, the robot will have to determine what composition of the controllers (indicated in the corresponding composed causal graph in Figure 3.11) to execute within the sequence by selecting the composed controller with the greatest predicted utility according to the policy in Equation 3.11.

3.4.1.3 Composed Causal Graphs

The set of causal graphs G_C indicate the composed effects of the controllers within the multi-objective `insert` and `screw` actions. As shown in Figure 3.11, the causal graphs indicate the controllers that are involved in these connect actions, the

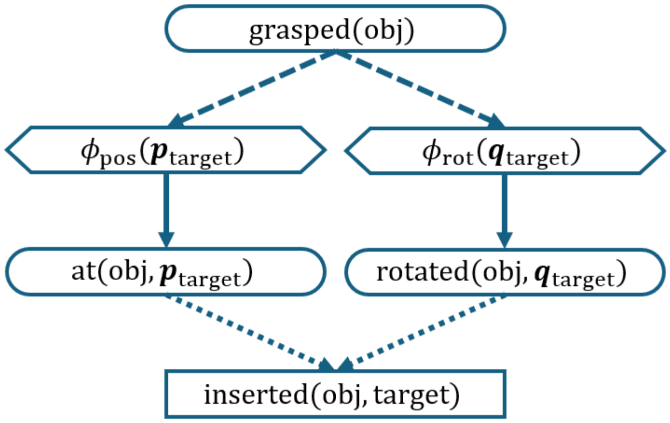
```
action-sequence-function insert(object=obj, target_object=target):
    effects: [inserted(obj, target)]
    return [
        execute(ϕ_6Dpose(ready(target))),
        execute(argmax_a(Q̂_insert(s,a))),
        connect(obj, target),
        execute(ϕ_gripper(open(ee)))
    ]
```

(a) Temporal action sequence function τ_{insert} for `insert` action. The action selected for the second step $\arg\max_a(\hat{Q}(s,a))$ is chosen from the possible compositions indicated in the composed causal graph in Figure 3.11a.

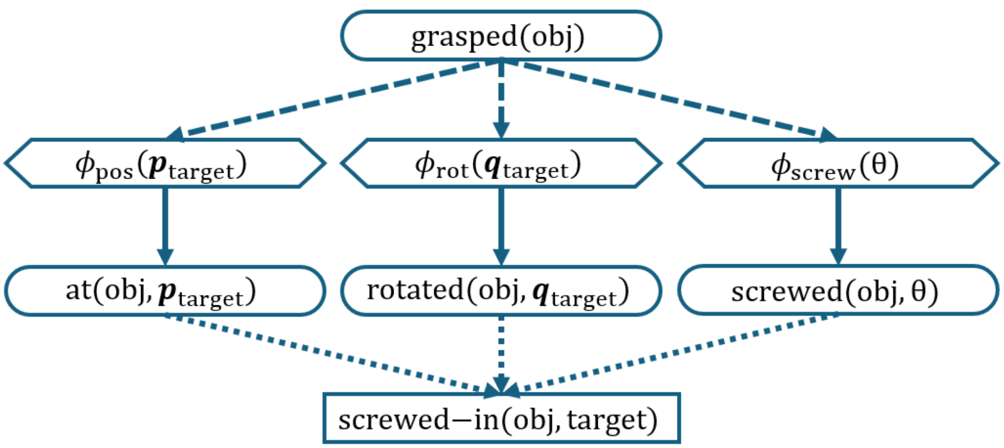
```
action-sequence-function screw(object=obj, target_object=target):
    effects: [screwed-in(obj, target)]
    return [
        execute(ϕ_6Dpose(ready(target))),
        execute(argmax_a(Q̂_screw(s,a))),
        connect(obj, target),
        execute(ϕ_gripper(open(ee)))
    ]
```

(b) Temporal action sequence function τ_{screw} for `screw` action. The action selected for the second step $\arg\max_a(\hat{Q}(s,a))$ is chosen from the possible compositions indicated in the composed causal graph in Figure 3.11b.

Figure 3.10: The temporal action sequence functions τ_{insert} and τ_{screw} for the high-level `insert` and `screw` actions. These action sequences connect an object `obj` to a target object `target` by executing the controller composition with the highest predicted state-action utility. We assume we have a `ready` function that determines an initial pose for beginning the connection actions relative to the given `target`. Note that the `connect` action in these sequences attempts the simulated welding, and is only successful if the selected controller composition effectively achieves the composed effects required for part connection.



(a) Composed causal graph $G_{C,\mathrm{insert}}$ for achieving the composed effect `inserted`. Insertion is based on the relative positioning and rotation between the object parts. The composition with the highest predicted utility will be selected for execution in the temporal action sequence in Figure 3.10a.



(b) Composed causal graph $G_{C,\mathrm{screw}}$ for achieving the composed effect `screwed-in`. Screwing is based on the relative positioning and rotation between the object parts as well as the pressure and screw motion applied by the screw controller. The composition with the highest predicted utility will be selected for execution in the temporal action sequence in Figure 3.10b.

Figure 3.11: The composed causal graphs for insertion and screwing connection actions. Both connection actions have pre-conditions that the object `obj` is grasped by the robot. Grasping is a pre-condition for executing the appropriate controllers to achieve the intended composed effects.

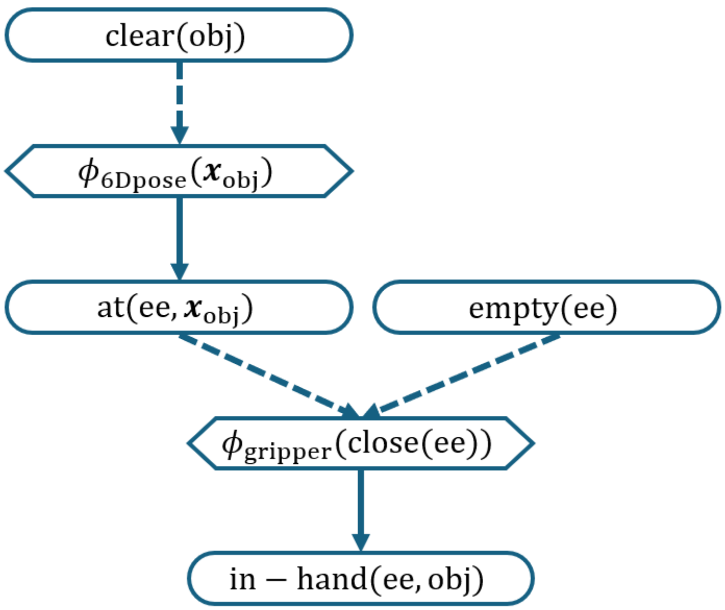


Figure 3.12: The composed causal graph $G_{C,\text{pickup}}$ for achieving effect `in-hand`. Since picking up an object does not play a multi-objective role in the task plan, the causal graph $G_{C,\text{pickup}}$ does not include any composed effect nodes. Instead, this causal graph shows a causal chain leading up to achieving effect `in-hand`. Since no two controller nodes exist together on any level of the graph, the robot does not need to estimate the state-action utility of any composed controller behaviors.

Executed Controller Composition $a = \texttt{execute}(\phi)$	Predicted State-Action Utility $\hat{Q}(s, a)$
$\phi_{\text{pos}} \triangleleft \phi_{\text{rot}}$	**0.723**
$\phi_{\text{rot}} \triangleleft \phi_{\text{pos}}$	0.711

Table 3.1: Utility predictions for `insert` action, computed according to Equation 3.10 with $N = 500$ Monte Carlo simulations for each possible composition ϕ.

pre-conditions of enacting these compositions, and the intended composed effects of these compositions. The robot will use the utility predictions from the *causal control basis* to determine how to compose these controllers together.

We also include causal graphs for actions that do not require composed controllers, such as the `pick-up` action in Figure 3.12. While this causal graph does not indicate the need for any controller compositions, it contains information about the relevant controllers, pre-conditions, and intended effects required to perform the `pick-up` action.

3.4.2 Composed Causality Predictions

For the `insert` and `screw` connection actions, the robot simulated $N = 500$ executions of each possible composition. We used time threshold $T = 300$ controller updates as the cutoff for the Monte Carlo simpulations. The predicted utility $\hat{Q}$ was computed as described in Section 3.3.3.

The state-action utility predictions for the `insert` and `screw` actions are shown in Table 3.1 and Table 3.2, respectively. In practice, expert users may define fixed controller compositions, regardless of the initial and next states, because the composed effects of these behaviors is expected to be the same regardless of the state [46, 47, 45, 48]. We hypothesize that the robot can similarly utilize fixed controller compositions, and therefore highlight the composition with the highest utility that will be executed in the assembly trials in Section 3.4.4.

The composition with the highest utility for the `insert` action indicates that

Executed Controller Composition $a = \texttt{execute}(\phi)$	Predicted State-Action Utility $\hat{Q}(s, a)$
$\phi_{\text{rot}} \lhd \phi_{\text{screw}} \lhd \phi_{\text{pos}}$	**0.937**
$\phi_{\text{pos}} \lhd \phi_{\text{screw}} \lhd \phi_{\text{rot}}$	0.936
$\phi_{\text{screw}} \lhd \phi_{\text{pos}} \lhd \phi_{\text{rot}}$	0.929
$\phi_{\text{pos}} \lhd \phi_{\text{rot}} \lhd \phi_{\text{screw}}$	0.925
$\phi_{\text{screw}} \lhd \phi_{\text{rot}} \lhd \phi_{\text{pos}}$	0.923
$\phi_{\text{rot}} \lhd \phi_{\text{pos}} \lhd \phi_{\text{screw}}$	0.904

Table 3.2: Utility predictions for `screw` action, computed according to Equation 3.10 with $N = 500$ Monte Carlo simulations for each possible composition ϕ.

positioning the object should be performed subject to aligning the object with the target, $\phi_{\text{pos}} \lhd \phi_{\text{rot}}$. The composition with the highest utility for the `screw` action indicates that aligning the object should be performed subject to screwing and positioning the object, $\phi_{\text{rot}} \lhd \phi_{\text{screw}} \lhd \phi_{\text{pos}}$.

3.4.3 Interpretation of Utility Predictions

We can see in Table 3.1 and Table 3.2 that the predicted utilities of executing the different compositions for each connection action are all very close. This may suggest that the choice of composition hardly matters. However, the large body of research exploring the many applications of composed controllers suggest that these compositions are meaningful [46, 45, 47, 48] and worth learning about autonomously through exploration [6]. Instead, it seems likely that the simulations themselves may obscure more meaningful differences. For example, while assembly tasks are challenging, IKEA furniture tends to be relatively straightforward to assemble, and these tasks are further simplified in simulation [9, 10]. Furthermore, it may be that the summation performed to aggregate observations about the Monte Carlo simulations in Equation 3.10 [339, 340] may obscure more meaningful differences about the behaviors of these controllers in different cases. For example, it may be possible that the composition $\phi_{\text{rot}} \lhd \phi_{\text{pos}}$ consistently makes bad progress because the high priority positioning controller bumps the target out of the way before alignment can be achieved. To demonstrate initial feasibility of our *causal control basis* approach, we considered very few cases when estimating the action utility according to Equation 3.8. It seems likely that additional cases are needed in order to properly evaluate performance and failure cases of these different controller compositions in the context of an assembly task.

3.4.4 Furniture Assembly Task Results

The composition with the maximum utility prediction is used to execute connection actions in a variety of furniture assembly tasks to test the accuracy of the estimated *composed causality* of the controllers. We tested the `insert` action within 10 trials of swivel chair assembly and tested the `screw` action within 10 trials of table assembly, where each trial started with the furniture pieces randomly positioned on the floor. All trials were executed on a simulated Baxter robot in the IKEA Furniture Assembly Environment [9, 10] simulator. Across all trials, we compute the average task time and success rates of the `pick-up` action, the multi-objective `insert` and `screw` actions, and the entire assembly task. Summary information for the swivel chair trials are in Table 3.3 and table trials are in Table 3.4. Table 3.5 shows summary information across all 20 assembly trials. See Appendix A for more detailed information about all assembly trials. Note that compared to the time we may expect a human to assemble a similar furniture item, the average task execution times (approximately 4.5 minutes for the swivel chair and approximately 8 minutes for the table) are not particularly long. This is due to the simplifying welding constraint used to connect object parts in the simulator.

When actions failed, it was often due to joint limits or local minima being reached, especially during the `pick-up` actions. This is due to the predefined grasp poses that we assume are provided to the robot—namely from the `pre-grasp`, `grasp`, and `post-grasp` functions used to sequence the `pick-up` actions, as seen in

`Pick-Up` Action Success Rate	0.606
`Insert` Action Success Rate	**0.714**
Swivel Chair Assembly Task Success Rate	1.000
Average Task Execution Time (s)	266.241

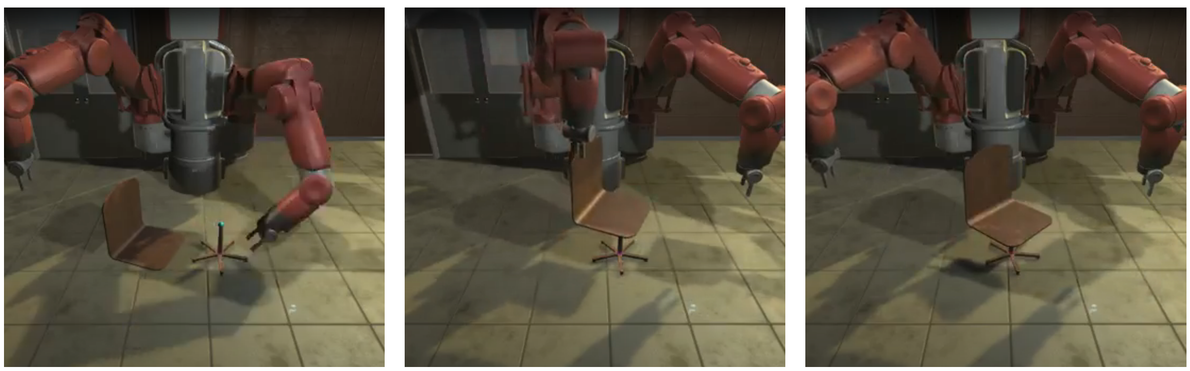
Table 3.3: Results from 10 swivel chair assembly tasks, with 33 `pick-up` attempts and 28 `insert` action attempts. The success rates of the actions with composed causality predictions from the *causal control basis* are highlighted to emphasize the performance in a full assembly task. Table A.1 contains more detailed information about all 10 swivel chair assembly trials.

`Pick-Up` Action Success Rate	0.909
`Screw` Action Success Rate	**0.952**
Table Assembly Task Success Rate	1.000
Average Task Execution Time (s)	492.072

Table 3.4: Results from 10 table assembly tasks, with 44 `pick-up` attempts and 42 `screw` action attempts. The success rates of the actions with composed causality predictions from the *causal control basis* are highlighted to emphasize the performance in a full assembly task. Table A.2 contains more detailed information about all 10 table assembly trials.

Figure 3.6. Selecting grasp poses is not the responsibility of our *causal control basis*; instead, we assume we have known grasp poses, similar to affordance templates [123]. As described in Section 3.3.4, the robot was allowed to retry an action whenever it failed. The success rates of the multi-objective connection actions reflect the performance of the *causal control basis*, and are highlighted in bold in Table 3.3 and Table 3.4. For 28 `insert` attempts across 10 swivel chair assembly trials, the `insert` action success rate was 0.714 and the task success rate was 1. For 42 `screw` attempts across 10 table assembly trials, the `screw` action success rate was 0.952 and the task success rate was 1. The high task success rate is a feature of the repeated action attempts described in Equation 3.12. The success rates for both actions in the swivel chair tasks are lower because the object parts required the robot to reach its arm much closer to the floor, and towards the limits of its reachable workspace. Images from three swivel chair trials are in Figure 3.13 and table trials are in Figure 3.14.

(a) Swivel Chair Assembly Trial 1.



(b) Swivel Chair Assembly Trial 2.

(c) Swivel Chair Assembly Trial 3.

Figure 3.13: Execution of three swivel chair assembly trials using multi-objective `insert` actions to connect parts together. Trials were executed on a simulated Baxter robot in the IKEA Furniture Assembly Environment [9, 10] simulator.

(a) Table Assembly Trial 1.

(b) Table Assembly Trial 2.

(c) Table Assembly Trial 3.

Figure 3.14: Execution of three table assembly trials using multi-objective `screw` actions to connect parts together. Trials were executed on a simulated Baxter robot in the IKEA Furniture Assembly Environment [9, 10] simulator.

Connection Action	**Successful Connections**	**Connection Attempts**	**Success Rate**
`Insert`	20	28	0.714
`Screw`	40	42	0.952
TOTAL	**60**	**70**	**0.857**

Table 3.5: Success rates of the multi-objective connection actions based on attempted and successful connection actions across all 20 assembly trials.

The similarity of the multi-objective connection action success rates (Table 3.3 and Table 3.4) and the predicted normalized state-action utilities (Table 3.1 and Table 3.2) indicates that the utility predictions accurately capture the effectiveness of performing the composed controllers during task execution. When the connect actions did not result in successful connections (due to joint limits or collisions between objects), the robot would retry the action. The task success rate for both the swivel chair and table tasks indicate that the robot was able to recover in these cases and achieve a successful connection.

3.5 Future Work

Additional evaluation is needed to test how the *causal control basis* predictions perform in real-world furniture assembly experiments. Real-world furniture assembly experiments would force us to relax assumptions in the simulated experiments, namely: (a) that the robot has perfect affordance-based perception of the objects in the scene; (b) that the robot has waypoints (similar to affordance templates) for picking up and preparing the objects for connection; and (c) that the test for secure part connection is achieved through a simulated welding constraint. Implementing our work on a real-world robot would involve perceiving the object parts and affordances; computing pre-grasp, grasp, and post-grasp poses relative to the objects during task execution; and making the connection actions more realistic. The connection actions in particular will be an interesting challenge, for example by considering depth of insertion and screwing. We suspect part connection may be a whole-body control problem, since the robot must maintain alignment and positioning of the object parts while performing the connection and testing for security based on applied forces. Real-world furniture assembly tasks will significantly improve the evaluation of our *causal control basis* methodology.

Future work includes assembling more furniture pieces, assembling furniture with more challenging initial part poses, and modifying the controller implementations for a real-world robot rather than in simulation. Future work could also include creating a more robust control basis by incorporating dynamics, including controller behaviors for avoiding joint limits and collisions between objects, and extending our *causal control basis* to coordinate the arms for bimanual or whole-body manipulation tasks.

The utility predictions from the *causal control basis* are based on the learning rules for Monte Carlo simulations. However, our results show evidence that more advanced analysis is required to appropriately characterize the behavior and failures of composed controllers. In particular, our methodology reasons about the composed controller as a whole, rather than observing progress made toward the individual controller objectives. Further data collection about the individual controllers could provide additional insights into trends that cause the composed controller to not achieve its composed effects.

The multi-objective examples presented in this chapter as well as controller compositions described in the literature [46, 45, 47, 48, 6] frequently consider compositions of two or three controllers. While theoretically, compositions could be over any number of controllers, there may be more practical limits to how many controllers are composed at any one time. For example, suppose a mobile manipulator robot has 3 controllers for the mobile base responsible for balancing, traveling (lateral movement), and turning (rotational movement) as well as 2 controllers for the manipulator arm responsible for positioning and aligning the end-effector. There likely would not be a case where the robot would need to compose all 5 controllers together, since the controllers affect different joint groups. Even within a single joint group, there is a limit to how many actions the robot can reasonably perform at once. So the robot may prioritize balancing above all else, then rotational movements for turning the mobile base and aligning the end-effector, and finally lowest priority are positioning movements for lateral travel and positioning the end-effector. These practical questions will help optimize the performance of multi-objective actions as research scales to more whole-body objectives.

3.6 Discussion and Conclusion

The robot's ability to successfully assemble different furniture pieces demonstrates the promise of our proposed *causal control basis* in predicting the composed effects of controller behaviors. The *causal control basis* effectively extends the principles of Semantic Robot Programming—that we need an intuitive way to declaratively program robots to perform tasks—into more challenging tasks that involve multi-objective actions. The *causal control basis* describes the intended results of the multi-objective actions, but leaves the robot to determine how to compose the appropriate controller behaviors and perform the action successfully. The *causal control basis* represents important information that users often already provide robots in some form. For example, the causal graphs encode information similar to symbolic descriptions of action pre-conditions and post-conditions and the temporal graphs encode information similar to hard-coded behavior sequences. Therefore, our *causal control basis* allows users to intuitively share knowledge about actions such that robots can autonomously determine how to enact those behaviors in challenging tasks.

In this work, we proposed a *causal control basis* for achieving *composable causality* reasoning. Our *causal control basis* allows the robot to predict the utility of executing a controller composition to achieve its composed effects, thereby estimating the composed causality of multi-objective actions. Our work in *composable causality* demonstrates that reasoning over a *causal control basis* provides the robot with the declarative knowledge necessary to autonomously compose controller behaviors without predefined priorities to achieve furniture assembly tasks.

3.7 Contribution to Dissertation Goals

In the context of this dissertation, the *causal control basis* contributes to the robot's abilities to perform complex manipulation tasks that involve multi-objective affordances on direct and indirect objects, such as assembly and tool-use tasks. Furthermore, our work in *composable causality* provides a framework for limiting the required expert knowledge engineering and expert programming by placing more responsibility on the robot to determine how to execute these complex multi-objective behaviors. The *causal control basis* improves the robot's reasoning capabilities in manipulation tasks, increases the robot's ability to autonomously compose controller behaviors, and reduces reliance on expert knowledge. The CURED Framework emphasizes the importance of understanding on human-robot teams, and reducing expert knowledge in favor of robot reasoning enables robots to effectively work alongside non-expert humans. In this way, *composable causality* reasoning not only makes the robot more capable of performing assembly and tool-use tasks, but also makes the robot more trustworthy, understandable, and explainable when acting in challenging problem domains.

3.8 Acknowledgments

This work was supported in part by NASA Space Technology Graduate Research Opportunity (NSTGRO) grant 80NSSC20K1200. Special thanks to Xiaotong Chen, Zhen Zeng, Kaizhi Zheng, Qiuyu Shi, and Odest Chadwicke Jenkins for their support and contributions to this project.

Chapter 4

Multi-Fingered End-Effector Grasp Reflex Modeling for One-Shot Tactile Servoing in Tool Manipulation Tasks

Autonomous tool manipulation tasks are challenging for robots because they must reason over the tool's object affordances, how to grasp the tool so it may be used, how the tool will interact with other objects in the environment, and how to perform the complex tool affordances to complete the manipulation task. Focusing on tool grasping presents further challenges, specifically generalization to novel tools and modeling the problem in an explainable way suitable for safety-critical task domains, such as robots operating autonomously to perform repair tasks in NASA lunar habitats. In this work, we focus on grasping tools in an explainable way that can be generalized to novel tools. We present a logistic regression based *grasp reflex model*, which maps continuous end-effector sensor data to a set of discrete symbolic states. An adjustment policy uses these symbolic states to compute the appropriate gradient to change the end-effector pose and increase the probability of a secure tool grasp. Once the tool grasp is sufficiently secure, the robot proceeds with the rest of the manipulation task. We test our *grasp reflex model* on 6 novel tools, and find that the model achieves one-shot generalization by successfully using tactile servoing to secure grasps from one example of a secure grasp state. The robot's ability to learn to grasp tools in an explainable way that achieves one-shot generalization to novel tools demonstrates the power of our *grasp reflex model* in allowing robots to achieve autonomous tool manipulation tasks.

4.1 Introduction

As robots become more capable and take on more tasks, autonomous tool manipulation will be an important task for robots to accomplish. Not only do we want robots to be able to use tools, but we want robots to be able to generalize their knowledge and apply their learned behaviors to a wide variety of tools without significant retraining. Tool manipulation is of particular interest for robotics since it requires understanding of the tool's object affordances [33], grasping the tool intentionally so it may be used in the subsequent manipulation task, reasoning over object-centric interactions [5, 123] between the tool and other objects, and executing complex tool-use behaviors through composition [46, 45, 47, 48, 96] and sequencing [7, 186] of lower-level actions. In this work, we focus on learning to grasp tools in a way that generalizes to novel tools.

Autonomously grasping tools presents a number of challenges. Data-driven approaches show significant promise, but neural network based algorithms reduce the explainability of the learned model. While these algorithms have proven to generalize well to novel tools, inexplicable black box algorithms cannot be trusted in safety-critical task domains [71, 72, 73, 74]. The challenge we aim to address in this work is how to use a data-driven approach to learn to grasp tools using a *simple explainable model* that *generalizes to grasping novel tools* not present in the training data.

We take inspiration from a human grasp reflex by allowing the robot to adjust its grasp. As seen in Figure 4.1, the robot will attempt a grasp on the tool, validate the grasp using sensor data, and adjust the grasp as needed until the tool is securely grasped. By allowing for adjustments to the grasp, we simplify the learned model to map from sensory input to a small set of discrete states that can be adjusted to improve the grasp quality. This approach of adjusting the grasp

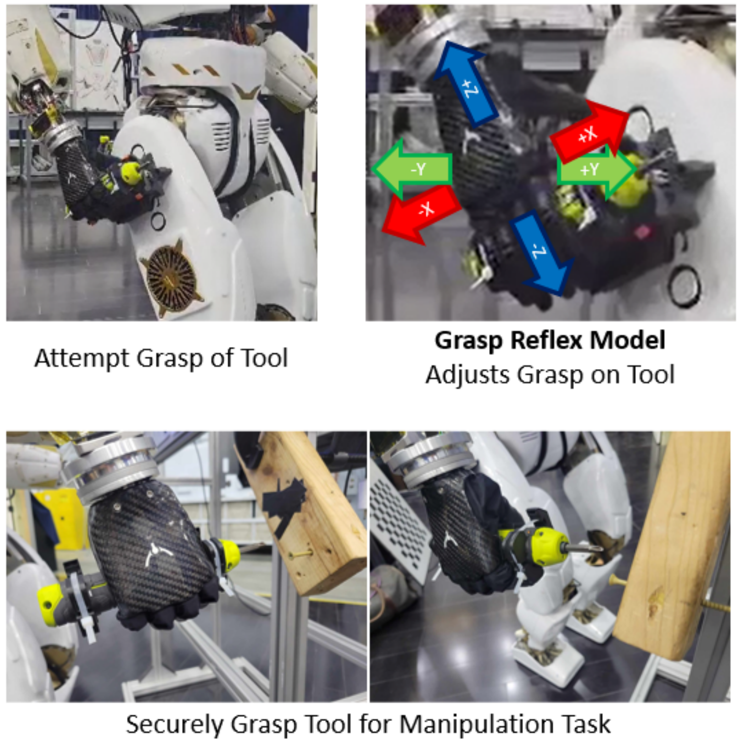


Figure 4.1: A robot using a *grasp reflex model* to iteratively adjust its grasp until a secure grasp is achieved.

to increase the probability of a secure grasp is tool-agnostic, and can generalize to unseen tools.

In this chapter, we present a logistic regression based ***grasp reflex model*** for multi-fingered end-effectors. Our *grasp reflex model* maps from multi-fingered end-effector sensor data to a set of discrete symbolic states. We use a policy that adjusts the grasp according to the detected symbolic state in order to increase the probability of a secure power grasp. Once the robot is reasonably confident the grasp around the tool is secure, it proceeds with the rest of the tool manipulation task. In constructing our *grasp reflex model*, logistic regression analysis identified statistically significant interactions between tool-agnostic model variables. We test the generalization of our model on a set of 6 novel tools, and find that our *grasp reflex model* achieves one-shot tactile servoing on these tools with a success rate of about 0.73. This success rate indicates that the *grasp reflex model* accurately captures grasping relationships for tools similar to the tool used for training, but may not sufficiently secure grasps on tools that are significantly different in size, shape, or weight from the training tool. Our work in multi-fingered end-effector grasp reflex modeling for one-shot tactile servoing demonstrates the power of training a simple explainable *grasp reflex model* that generalizes to novel tools and furthers robots' capabilities of autonomously performing tool manipulation tasks.

4.2 Related Work

4.2.1 Grasping in Object Manipulation Tasks

Robot grasping aims to restrain and manipulate objects, and comes with unique challenges of controlling an end-effector while achieving contacts with and applying forces to objects [199]. Many works have explored solving the problem of robot grasping. Approaches to grasping are analytic (geometric) or data-driven (empirical); model-based (which often involves pose estimation of known objects) or model-free (which aims for generalized grasps on novel objects) [207]; and discriminative (probabilistically ranking grasp candidates) [220], generative (generating grasp candidates) [342], or hybrid (combining modeling and learning techniques) [343]. Robot grasping can be performed autonomously, with real-time teleoperation from a human operator, or with autonomous assistance to the operator [205]. Grasping is highly dependent on the robot hardware, as gripper designs support different grasp configurations and affect how the robot will interact with objects [196, 344, 345, 197].

Grasping for tool manipulation tasks presents several unique challenges. Tool manipulation tasks require manipulation-oriented grasps [37], in which tools are grasped for use in further manipulation tasks [39]. Tool grasp requirements constrain grasping tasks such that finding optimal grasps can be formulated as a search problem [37]. We take inspiration from searching for grasps [37] by testing and verifying successful tool grasps with sensory feedback [229].

4.2.2 Tactile Servoing

Because we expect the robot to attempt tool grasps and verify the grasp based on sensor data, we consider *tactile servoing* methods. A counterpart to visual servoing, tactile servoing uses data from tactile or contact sensors to control a robot [230]. Tactile servoing is essential for autonomous dexterous manipulation and in-hand manipulation. Tactile servoing methods utilize the abundant sensory signals from arrays of tactile contact sensors to estimate contacts and forces between end-effectors and objects. These approaches typically rely on multi-fingered robot hands as opposed to grippers in order to provide more degrees of freedom for controlling the object and more sensory information [232].

Many works explore the variety of control tasks that can be achieved through tactile servoing [233, 234, 346]. Tactile servoing methods can be used for state estimation, such as estimating in-hand object poses [347] and external contact locations [348]. Reasoning over tactile contacts allows robots to perform in-hand object reconfiguration tasks [349]. Visual and tactile servoing can often work together, with vision information improving tactile servoing methods and tactile information improving visual servoing methods [236]. For example, exploratory tactile servoing or active touch can be applied to identifying object shapes through active perception [137]. Combining visual and tactile feedback has also been explored for tool grasping and manipulation tasks [229, 350, 351, 352].

Since tactile servoing methods typically rely on large amounts of data from tactile sensor arrays, many works utilize data-driven machine learning approaches. Several works utilize algorithms such as convolutional neural networks [237], deep learning neural networks [238], or offline neural network based learning [239]. While these algorithms have proven effective, these black box models reduce explainability, which can be problematic for safety-critical problem domains [71, 72, 73, 74]. In this work, we take inspiration from data-driven learning from demonstration approaches [239] but aim to process our sensor data using a more explainable model for use in safety-critical applications.

4.2.3 Human-Inspired Grasp Reflex

The human grasp reflex has been well studied in developmental psychology. In newborn humans, the palmar reflex is an involuntary response in which babies close their fingers around an object when the palm is touched [353]. This reflex disappears at a young age when babies are able to voluntarily use their hands and grasp objects [354, 355].

Since tactile servoing often relies on multi-fingered robot end-effectors, many works turn to biologically-inspired and human-inspired multi-fingered grasps. To mimic the vast amount of sensory information provided by human hands, some works outfit robot hands with large arrays of tactile contact sensors [231]. Many works explore training robot end-effectors to mimic human grasp reflexes to assist people who use prosthetic hands [356] and aid people in recovering from paralysis through robotic rehabilitation [357].

Several works take biological inspiration for the algorithms themselves. For example, one work takes inspiration from human grasps and assumes uniform contact between all fingers and the object is required for a secure grasp [198] rather than exploring the true model of the robot grasp. Tomovic *et al.* [206] attempt to geometrically model the object being grasped using a number of geometric primitives in order to simplify a human-inspired generalized grasp. Deckers *et al.* [358] demonstrate that robots can learn to grasp objects based on proprioceptive information from a gripper using reinforcement learning. Another work models a grasp reflex using biological neuron models [223] as opposed to the artificial neuron models in neural networks. Tieck *et al.* [209] propose a biologically-inspired spiking neural network, which achieves one-shot learning on sphere, cylinder, and pinch grasp primitives.

While these works demonstrate the power of biologically-inspired algorithms, we aim to explore simpler more explainable algorithms for a human-inspired robotic grasp reflex.

Rather than imitating biological systems, *artificial reflex control (ARC)* aims to create an analogous approach [224], resulting in a simpler more explainable reflex model. ARCs analyze sensory data to learn patterns of response. ARCs emphasize the importance of responding to sensory information, and map sensory patterns to appropriate joint states to create the reflex behavior [224]. He *et al.* [225] apply ARCs to demonstrate an open/close gripper reflex that responds to pressure, similar to a human baby's pressure-based grasp reflex. We take inspiration from ARC principles and aim to apply ARCs to more complex tool manipulation tasks. We also aim to learn the mapping function from sensory inputs to joint states to reduce knowledge engineering and hand-written rules required in past ARC implementations [224].

4.3 Methods

4.3.1 Problem Formulation

To perform tool manipulation tasks, the robot must first securely grasp the tool. We aim to model a grasp reflex that guides the robot's fingered end-effector to achieve a secure grasp. Due to uncertain nondeterministic robot motion, we expect the robot to attempt a grasp, detect the state of the fingered end-effector relative to the tool, and adjust the relative poses and configurations until the grasp is secure.

The end-effector state at time t is represented by the end-effector pose $\mathbf{x}_t \in SE(3)$ and the end-effector joint configuration $\mathbf{q}_t \in C$ for configuration space C. Let $\texttt{secure} : SE(3) \times C \rightarrow \{T, F\}$ be a predicate that determines if the continuous end-effector state is securely grasping a tool. This predicate function could be human-defined or learned, and may be based on the magnitude of forces tolerated by the end-effector before dropping the grasped tool or an invariant of the relative poses between the end-effector and tool when the robot moves. Put another way, when the tool can be considered part of the robot's "self" by moving with the robot's end-effector, it is securely grasped.

To achieve a secure grasp, we assume we have a set of actions $A = \{a_1, \cdots, a_n\}$ such that each action represents a translational or rotational control adjustment along one of the end-effector basis directions. At time t, we aim to express the continuous end-effector state $\mathbf{x}_t, \mathbf{q}_t$ as a discrete symbolic state s_t. The symbolic states $s_i \in S$ are defined as the prerequisite states for the actions $a_i \in A$, such that each symbolic state represents a disjoint region of the continuous state space, indicating the appropriate grasp adjustment action. Once the symbolic state s_t is identified, the robot can adjust its grasp on the tool using the policy $\pi : S \to A$, where $\pi(s_i) = a_i$ for all $1 \le i \le n$.

After making a sequence of adjustment actions $a_1, \cdots, a_t$, the robot will have secured a grasp on the tool when the robot is reasonably certain about its state prediction—meaning the predicted probability that the end-effector is in a given state is above probability threshold p^*—and the corresponding continuous end-effector state is secure. The robot has achieved a secure grasp at time t when the condition:

$$\hat{P}(s_t) > p^* \quad \bigwedge \quad \texttt{secure}(\mathbf{x}_t, \mathbf{q}_t) \tag{4.1}$$

is true in the current state.

4.3.2 Problem Statement

The goal of our work is to model the unknown function f that maps end-effector information $\mathbf{x}_t, \mathbf{q}_t$ and a reference secure grasp configuration $\mathbf{q}^* \in C$ for the tool being grasped to one of the symbolic states $s_i \in S$ with known adjustment actions

$a_i \in A$. In particular, we need to learn the function $f : SE(3) \times C \times C \to \mathbb{R}^n$, which maps end-effector state information and a reference secure grasp configuration to the *probability predictions* for each symbolic state. Let:

$$f(\mathbf{x}_t, \mathbf{q}_t, \mathbf{q}^*) = \mathbf{v}_t = \begin{bmatrix} v_{t,1} \\ \vdots \\ v_{t,n} \end{bmatrix} = \begin{bmatrix} \hat{P}_t(s_1) \\ \vdots \\ \hat{P}_t(s_n) \end{bmatrix} \tag{4.2}$$

where the elements of output vector $\mathbf{v}_t$ give probability predictions for symbolic states $s_1, \cdots, s_n$. Please see Section 4.3.3.1 and Section 4.3.3.2 below for discussion on the implications of this model formulation. The model f classifies end-effector state information at time t into symbolic states as:

$$s_t = \arg\max(\mathbf{v}_t) = \arg\max_{1 \leq j \leq n}(v_{t,j}) \tag{4.3}$$

which selects the state with the greatest predicted probability. This state classification s_t is fed into the policy to determine the adjustment action $a_t = \pi(s_t)$. The grasp is secured at time t when the model f predicts the probability for state s_t is above probability threshold p^* and the corresponding end-effector state is determined to be secure. The robot's grasp is secure at time t when the continuous end-effector state $\mathbf{x}_t, \mathbf{q}_t$ predicted to be represented by symbolic state s_t as in Equation 4.2 satisfies the secure grasp condition in Equation 4.1.

4.3.3 Notes on Problem Statement

In the following sections, we comment on some implications and nuances of the model formulation in Equation 4.2.

4.3.3.1 Mapping Continuous End-Effector States to Discrete Symbolic Spaces

Note that our problem statement distinguishes between the use of continuous and discrete state spaces to describe the state of the end-effector grasp. The security of a grasp is determined by the magnitude of forces that can be tolerated by the current state of the end-effector without changing the relative pose between the end-effector and the grasped object. For this reason, the `secure` predicate is defined on the *continuous* end-effector state space $SE(3) \times C$ that includes the end-effector pose and configuration. The model f maps this continuous state information to *discrete* symbolic states representing *adjustments* the robot can make in order to improve the security of the grasp.

4.3.3.2 Properties of Symbolic Grasp States

Note that symbolic states $s_i \in S$ do not necessarily cover the full state space of end-effector configurations, meaning $\sum_{j=1}^{n} \hat{P}_t(s_j) \neq 1.0$. Our only assumption is that the continuous end-effector state space contains some secure grasp states and adjustable states. The secure grasp states are detected by predicate `secure`. Adjustable end-effector states can be characterized by each of the symbolic states $s_i \in S$, which determine the appropriate adjustment action $a_i \in A$. However, the symbolic states S need not represent every possible state, just the states we expect the robot to be able to identify or adjust.

Our assumption that the symbolic states do not necessarily cover the state space means that the symbolic states $s_i \in S$ need not be mutually exclusive nor exhaustive. As an example for how symbolic states may not be mutually exclusive, we note in Section 4.6 that states may be adjusted using linear combinations of adjustment actions, which means a single end-effector state may be represented by multiple symbolic states at once. As an example for how symbolic states may not be exhaustive, we show in Section 4.4.1.3 that our implementation includes a generic symbolic state that captures when the robot does not have sufficient information to confidently adjust its grasp. These properties of symbolic states avoids overtrusting the human-robot team to exhaustively explore the space of end-effector configurations.

4.3.4 Grasp Reflex Model

We propose a *grasp reflex model*, which models function $\mathbf{v}_t = f(\mathbf{x}_t, \mathbf{q}_t, \mathbf{q}^*)$ in Equation 4.2 as a *logistic regression model.* For some reference tool, a reference secure grasp joint configuration $\mathbf{q}^*$ should be recorded, either with assistance from a human operator or learned from experience. The n basis adjustment actions and corresponding symbolic states will be defined for the robot's fingered end-effector. The definitions used for our implementation are described in Section 4.4.1. To train the *grasp reflex model*, data must be collected demonstrating examples of each of the n symbolic states, possibly hand labeled through teleoperation of the robot. For each example, end-effector pose and configuration information should be recorded for the best modeling.

Performing a logistic regression analysis on all collected variables as well as interactions between variables will identify which proprioceptive sensor data is most relevant to identifying the symbolic states for the fingered end-effector. Once statistically significant variables are identified, the final *grasp reflex model* can be trained and validated.

Note that validation accuracy of the *grasp reflex model* will quantitatively evaluate the model's accuracy in predicting the symbolic states according to the hand labeled training data, and will only give a small indication of the performance. Full evaluation is determined when the *grasp reflex model* is deployed on the robot and used to direct the robot to secure grasps on different tools, as described in Section 4.4.3.

Figure 4.2: NASA Johnson Space Center's Valkyrie robot (left) with the PSYONIC Ability Hand mounted (center) and a tool holster on the right hip (right).

4.4 Experiments and Results

We aimed to model the grasp reflex for a PSYONIC Ability Hand [359, 360] mounted on NASA Johnson Space Center's Valkyrie robot [92, 361, 245]. Valkyrie uses ROS1 [362] MoveIt [363, 364] for motion planning and IHMC Open Robotics Software [365] for whole-body control. The PSYONIC hand has 6 joints: one for each of the 4 fingers and 2 in the thumb (roll and pitch). We wanted Valkyrie to carry tools onboard, so Valkyrie was outfitted with a tool holster on its right hip. Since the tool holster is rigidly attached to the robot's hip, it was added to the robot model, described in the Unified Robot Description Format (URDF) [366, 367, 368]. The holster's pose was used as the initial guess for all autonomous grasp attempts, but the actual pose of the tool is not known. Due to uncertainty and nondeterminism in the robot motion, the robot needs to use tactile servoing to adjust its grasp from this initial guess. Figure 4.2 shows our experimental setup.

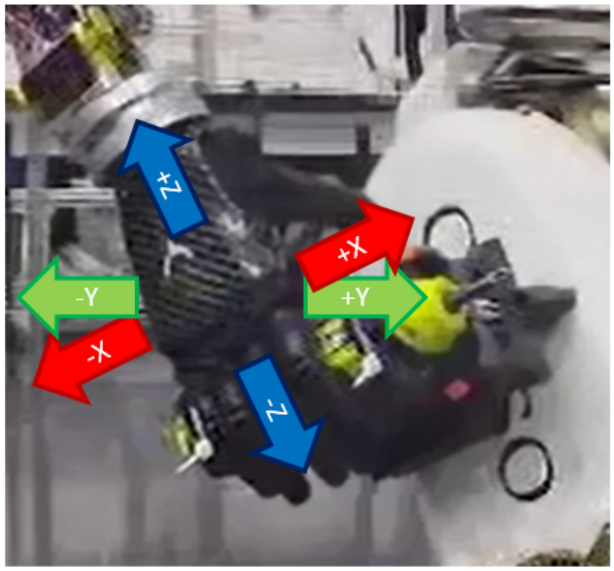


Figure 4.3: The action space is defined by small translational offsets along each of the end-effector x, y, z basis directions.

4.4.1 Action and State Space Definitions

4.4.1.1 Basis Adjustment Actions

The fingers of the PSYONIC hand are strong enough that small rotational offsets did not affect the success of the grasp. If the hand's rotation varied slightly from the rotation of the tool, the fingers pulled the hand's rotation into alignment with the tool when it attempted a grasp. For this reason, we considered only positive and negative translational offsets along the end-effector x, y, z basis directions for our action space, as seen in Figure 4.3. Each translational offset corresponds to an adjustment δ of the end-effector pose $\mathbf{x}_t$ along one of the basis directions. For our implementation, we used $\delta = 0.03$ meters along the basis directions. We found that this offset resulted in small enough adjustments as to not overshoot a secure grasp, which could lead to a local minimum when searching for a secure grasp.

4.4.1.2 Symbolic States

Each action in the action space corresponds to a symbolic state that will be identified based on our *grasp reflex model* prediction as in Equation 4.3. These symbolic states indicate the pose of the end-effector relative to the tool being grasped. We refer to these symbolic states using intuitive colloquial terms, for example "too far" or "too high".

As a proof-of-concept for our *grasp reflex modeling* approach to touch-driven control for tool grasps, we do not solely rely on the `secure` predicate function and instead attempt to characterize the range of possible secure grasps as symbolic states. We collectively refer to these secure grasps as a proper subset of our symbolic state space, $S_{\text{secure\_grasp}} \subset S$. We found it useful to denote different secure states, for example when the thumb is securely grasping the tool, when 2 or more fingers are securely grasping the tool, etc. The symbolic states do not provide any criteria for grasp security, however we gathered training data for different variations of secure grasps.

Note that the colloquial labeling of and differentiation between these symbolic states is ultimately up to the discretion of the operators based on the particular fingered end-effector and robot hardware in use. The only requirement for our method is that there are n symbolic states corresponding to adjustments that can be made by the n basis actions.

4.4.1.3 Grasp Adjustment Policy

Based on the action and state space definitions provided above, we define the following policy (visualized in Figure 4.3) for adjusting the PSYONIC hand's pose

$\mathbf{x}_t$ relative to the tool in state s_t:

$$a_t = \pi(s_t) = \begin{cases} \mathbf{x}_t & s_t = s_{\text{no\_contact}} \\ \mathbf{x}_t + \delta[1,0,0] & s_t = s_{\text{too\_backward}} \\ \mathbf{x}_t - \delta[1,0,0] & s_t = s_{\text{too\_forward}} \\ \mathbf{x}_t + \delta[0,1,0] & s_t = s_{\text{too\_far}} \\ \mathbf{x}_t - \delta[0,1,0] & s_t = s_{\text{too\_close}} \\ \mathbf{x}_t + \delta[0,0,1] & s_t = s_{\text{too\_low}} \\ \mathbf{x}_t - \delta[0,0,1] & s_t = s_{\text{too\_high}} \\ \mathbf{x}_t & s_t \in S_{\text{secure\_grasp}} \end{cases} \tag{4.4}$$

Note that the subset $S_{\text{secure\_grasp}}$ covers all of the different secure states mentioned in Section 4.4.1.2. Also note that state $s_{\text{no\_contact}}$ cannot be adjusted. However, it is important to include this state in the *grasp reflex model* to identify when not enough information is available to adjust the grasp.

4.4.2 Trained Grasp Reflex Model

We selected a drill as the reference tool to train our *grasp reflex model*, as seen in Figure 4.4. This tool was selected because it is symmetrical, relatively easy for the PSYONIC hand to grasp, and not too heavy. By placing the tool in the robot's hand, we acquired a reference joint configuration $\mathbf{q}^*_{\text{drill}}$ for a secured grasp around this drill.

For each symbolic state in the state space, we teleoperated the robot to 10 examples of that state. At each of these states, we recorded 10 data points in order to characterize sensor noise. This gave us 100 data points per symbolic state to train our *grasp reflex model*. For each data point, we recorded data on the

Figure 4.4: Set of tools used for experiments, including the reference drill (top) used for training. The novel tool instances (bottom) from left to right: compressed air can, selfie stick, screwdriver, level, gyroscopic drill, and paint scraper.

commanded and actual end-effector poses, commanded and actual joint states, and distances of each joint from the reference secured joint configuration. This data was split into training and validation sets for the construction of our *grasp reflex model*.

We performed a logistic regression analysis to identify relevant variables and to explore interactions between variables. This analysis identified the following variables as statistically significant (with p-value $p < 0.01$ for each variable's effect on the predicted state):

- Interaction between (multiplication of) the index and pinky distances between reference secure configuration and current configuration

- Interaction between (multiplication of) the middle and pinky distances between reference secure configuration and current configuration

Our final trained *grasp reflex model* $\hat{f}$ is the logistic regression approximation of Equation 4.2. Let $d_j = \|q_j^* - q_j\|$ be the distance between the reference secure and current configuration for end-effector joint j. Based on our logistic regression analysis, the final form for our learned *grasp reflex model* $\hat{f}$ is $\hat{f}(\mathbf{x}_t, \mathbf{q}_t, \mathbf{q}^*) = \frac{1}{1+e^{-\mathbf{u}_t}}$ where:

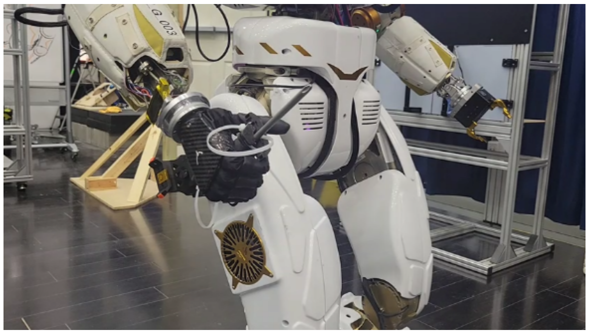
$$\mathbf{u}_t = \mathbf{c}_0 + \mathbf{c}_1(d_{\text{index}} \cdot d_{\text{pinky}}) + \mathbf{c}_2(d_{\text{middle}} \cdot d_{\text{pinky}}) \tag{4.5}$$

with learned model coefficients $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2$. Interestingly, the statistically significant variables indicate that not all fingers are required for the end-effector to securely grasp the tool, indicating redundancy in our fingered end-effector. The logistic regression analysis also indicated that the model is statistically significant with p-value $p \ll 0.001$. This indicates that we reject the null hypothesis, and there is a significant relationship between the identified variables and the symbolic grasp states. When validating the accuracy of this model against the validation data, the *grasp reflex model* achieved about 89% accuracy in identifying the symbolic grasp states.

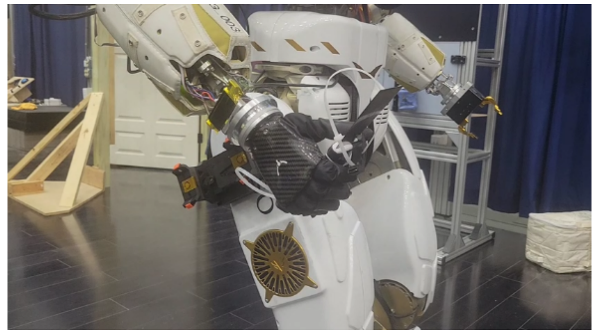
4.4.3 Generalization to Novel Tool Instances

To test the *grasp reflex model*'s ability to secure tool grasps in real-world tool manipulation tasks, we deployed our final trained model $\hat{f}$ on the Valkyrie robot. We wanted to test the *grasp reflex model*'s ability to secure grasps on both the drill used for initial model training, as well as novel tools. We considered a set of 6 novel tools, as seen in Figure 4.4.

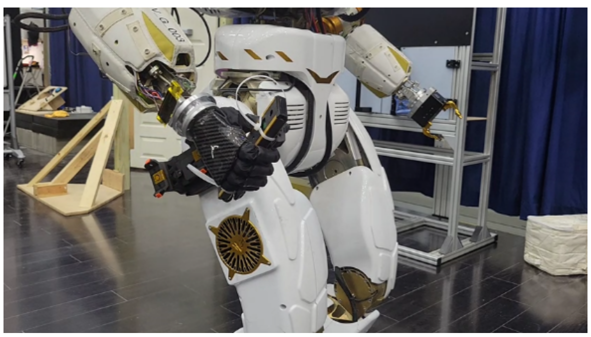
For each tool k, we placed the tool in the robot's hand and acquired a single reference joint configuration $\mathbf{q}_k^*$ for a secure power grasp around the tool. Note that this single reference secure configuration for each tool means we are asking our trained *grasp reflex model* $\hat{f}$ to perform *one-shot tactile servoing* on novel tools. To secure a grasp on tool k at time t, the robot used our *grasp reflex model* $\hat{f}$, current end-effector state $\mathbf{x}_t, \mathbf{q}_t$, and reference secure joint configuration $\mathbf{q}_k^*$ to predict the symbolic state $s_t = \arg\max(\hat{f}(\mathbf{x}_t, \mathbf{q}_t, \mathbf{q}_k^*))$. This state was then fed to the policy function π (Equation 4.4) to find the action $a_t = \pi(s_t)$ that would allow the



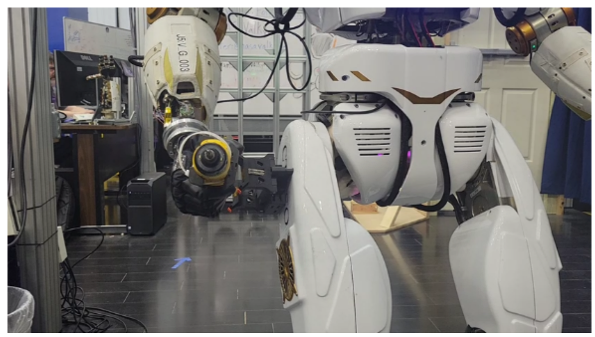
(a) Screwdriver:
SUCCESS



(b) Paint Scraper:
SUCCESS

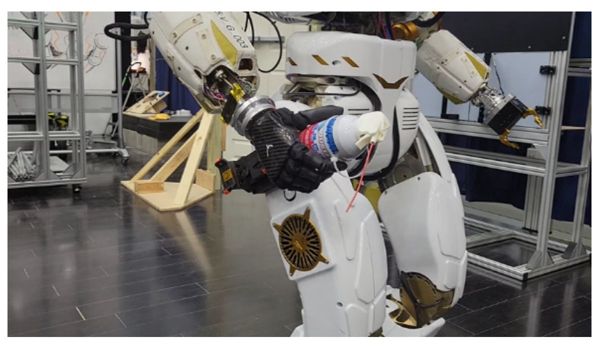


(c) Level:
SUCCESS



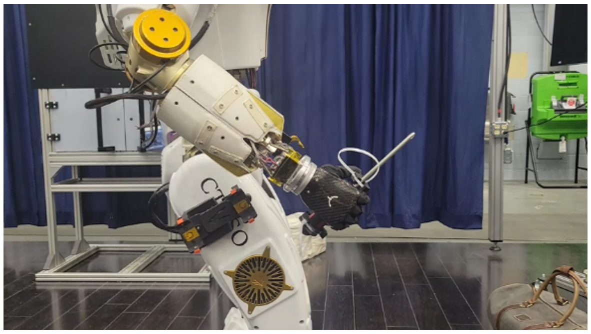
(d) Gyroscopic Drill:
FAILED

(e) Selfie Stick:
FAILED

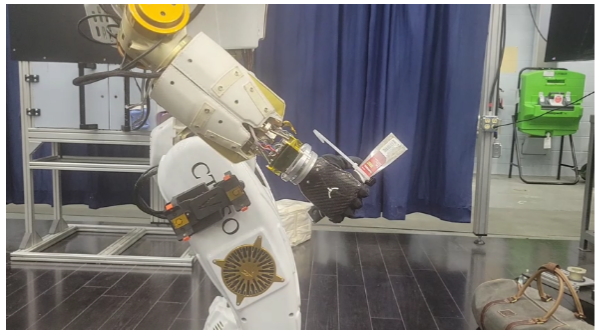


(f) Air Can:
FAILED

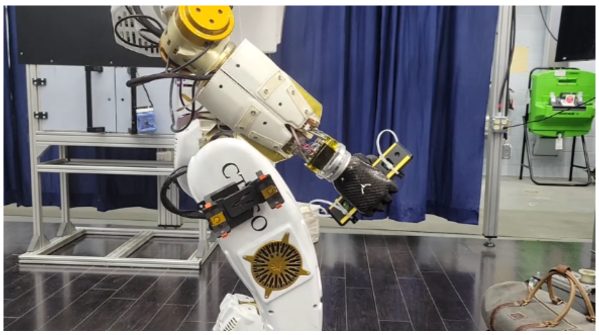
Figure 4.5: Random front-view trials of the *grasp reflex model* for one-shot tactile servoing on novel tools. Note that in all trials, in-hand grasps were achieved, but manipulation grasps may or may not have been achieved. The manipulation grasp success is indicated below each trial image.



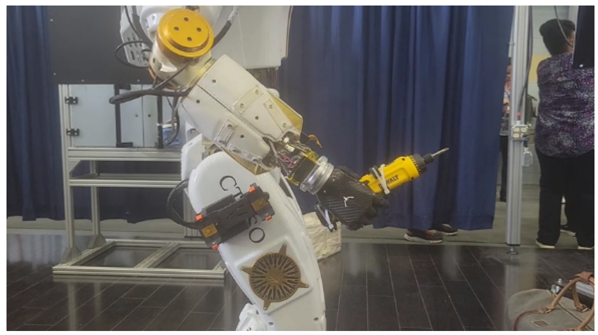
(a) Screwdriver:
SUCCESS



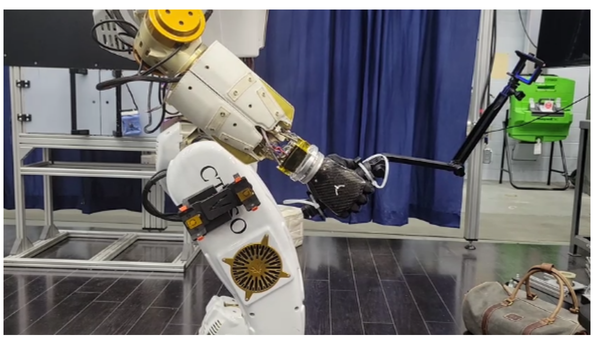
(b) Paint Scraper:
FAILED



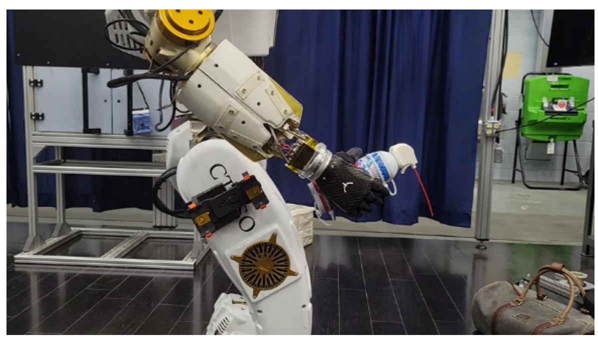
(c) Level:
FAILED



(d) Gyroscopic Drill:
SUCCESS



(e) Selfie Stick:
SUCCESS



(f) Air Can:
SUCCESS

Figure 4.6: Random side-view trials of the *grasp reflex model* for one-shot tactile servoing on novel tools. Note that in all trials, in-hand grasps were achieved, but manipulation grasps may or may not have been achieved. The manipulation grasp success is indicated below each trial image.

Tool	Practical for End-Effector	In-Hand Grasp Success Rate	Manipulation Grasp Success Rate
Drill	Yes	1.00	1.00
Screwdriver	Yes	1.00	0.83
Paint Scraper	Yes	1.00	0.67
Level	Yes	0.83	0.67
Gyroscopic Drill	Yes	1.00	0.50
Selfie Stick	**No**	1.00	0.33
Compressed Air Can	**No**	1.00	0.17
TOTAL	**-**	**0.98**	**0.60**
PRACTICAL TOTAL	**-**	**0.97**	**0.73**

Table 4.1: Results across 42 tool manipulation trials, with 6 trials per tool. We consider in-hand grasps secure if the robot did not drop the tool, and manipulation grasps secure if a human operator could not pull the tool from the robot's hand. The drill performs the best, since this tool was used to train the *grasp reflex model*. Other tools are more challenging since they differ from the drill in terms of size (paint scraper), graspable surface area (level), and/or weight distribution (front-heavy gyroscopic drill). Note that the selfie stick and compressed air can are not practical for the given end-effector, as they represent the lower and upper limits, respectively, of what can reasonably be grasped by the PSYONIC hand. More detailed information about the in-hand and manipulation grasps achieved in each trial can be found in Table B.1 and Table B.2, respectively.

robot to adjust its grasp on the tool and increase the model's predicted probability of a secure grasp. The robot would continue adjusting its grasp based on the *grasp reflex model*'s predictions indefinitely until the grasp was sufficiently secure. When the *grasp reflex model* predicted that the current state at time t was secure $\hat{P}_t(s_t) > p^* \ \bigwedge\ \texttt{secure}(\mathbf{x}_t, \mathbf{q}_t)$ according to the secure grasp condition in Equation 4.1, the robot would pull the tool from the tool holster and continue with the tool manipulation task. We used an intentionally low probability threshold $p^* = 0.10$ for detecting secure grasps to stress test the *grasp reflex model*'s ability to accurately detect states in the context of tool manipulation tasks.

For each tool, we performed 6 trials, with each trial beginning with a different initial guess on the grasp; noise was added to the tool holster pose to alter this initial guess, so each trial required at least 1 adjustment action before securing the grasp. For each trial, we recorded whether the grasp resulted in a secure *in-hand grasp* and a secure *manipulation grasp*. Here, we consider in-hand grasps secure if the robot successfully pulled the tool from the holster without dropping it to the ground. We consider manipulation grasps secure if a human operator could not pull the tool from the robot's hand without backdriving the PSYONIC finger joints. The resulting grasp success rates for each individual tool as well as the cumulative grasp success rates are shown in Table 4.1. See Appendix B for more detailed information about all grasping trials. Examples of one-shot tactile servoing instances for each of the 6 novel tools can be seen in Figure 4.5 and Figure 4.6.

4.4.4 Practical Interpretation of Grasp Reflex Model

The set of 6 novel tools was selected to stress test how well the *grasp reflex model* generalizes to novel tools. The novel tools were selected because they differed from the reference drill in size, graspable surface area, symmetry, weight, and/or weight distribution. In testing these tools, we found hardware limitations of the specific end-effector we modeled. In particular, the compressed air can and selfie stick seemed to mark the upper and lower limits, respectively, of how big around an object could be for the PSYONIC to securely grasp. Even with help from a human operator, it was difficult to obtain the reference secure joint configurations $\mathbf{q}^*_{\text{air\_can}}$ and $\mathbf{q}^*_{\text{selfie\_stick}}$ for a secure power grasp around these tools. Due to the hardware limitations, these objects are likely not practical for the end-effector to reliably manipulate in tool manipulation tasks. For this reason, Table 4.1 notes both the total cumulative in-hand and manipulation success rates for all 7 tools as well as the total *practical* cumulative in-hand and manipulation success rates for the 5 practical tools.

Even when considering the impractical tools, the *grasp reflex model* achieves manipulation grasps with a success rate of 0.60, indicating that the *grasp reflex model* effectively models the symbolic grasp states for the end-effector and generalizes to novel tools, achieving secure grasps more often than not. However, when we consider only the tools that can be *practically* manipulated given the end-effector hardware, the *grasp reflex model* achieves a higher manipulation grasp success rate of 0.73. The success rates for manipulation grasps indicate that while the *grasp reflex model* achieves manipulation grasps most of the time, the model could be improved by accounting for differences between the training tool and the novel test tools. In particular, the *grasp reflex model* struggled to achieve manipulation grasps on tools that were significantly different from the training tool in size (compressed air can), shape (selfie stick), and weight or weight distribution (front-heavy gyroscopic drill). The variation in these tools from the training tool indicate that the *grasp reflex modeling* approach could be improved by learning a distribution over relevant tool features and classifying how these features lend themselves to different types of grasps, such as precision or trigger grasps.

Since our definition of manipulation grasp is much more constrained than our definition of in-hand grasp, we expect manipulation grasps to be generally more difficult to achieve. Regardless of whether we consider the practical tools or all tools, the *grasp reflex model* achieves in-hand grasps with a success rate of over 0.97. This indicates that depending on how secure a grasp is required, our *grasp reflex model* achieves secure grasps with high fidelity without hand-tuned parameters or re-tuned parameters for generalization to different tools. Our experiment results provide evidence that our *grasp reflex model* balances performance and explainability in tool-use tasks, especially in safety-critical problem domains where explainability is necessary [71, 72, 73, 74].

4.5 Future Work

Future work includes extending to a larger action space, such as rotational adjustments and linear combinations of adjustments along the end-effector basis directions. Training the *grasp reflex model* over a set of representative tools (rather than just a single reference tool) could improve grasp success rates by modeling a distribution over challenging features such as tool size and weight distribution. This grasp reflex modeling approach could be generalized to different types of tool grasps—such as precision grasps or trigger grasps—rather than just the single overhand power grasp modeled in this work. We found that stress testing the *grasp reflex model* exposed hardware limitations of the end-effector, and further modeling and stress testing could improve our understanding of the practical capabilities of a given end-effector.

The implementation of the *grasp reflex modeling* approach presented in this chapter is a human-based approach, which depends on human-supplied information such as the selection of the tool, reference secure grasp configuration, basis adjustment actions, and corresponding symbolic states. Future research could learn all of this information through autonomous exploration or "play" with grasping different tools. Autonomous exploration of grasping and usage of the tools may allow the robot to move beyond needing a reference secure grasp configuration for each tool, and instead use tactile and force-torque information from the end-effector to reactively assess grasp security (effectively transitioning from one-shot to zero-shot tactile servoing). More generally, the robot could learn from experience about each object it grasps, such as learning the end-effector-specific `secure` predicate function, learning the reference close configuration for each tool instead of relying on the human-provided configuration, learning how combinations of primitive adjustment actions may improve grasp security, or learning from failure cases to improve performance. The goal of the work presented in this chapter is to establish that the *grasp reflex modeling* framework feasibly allows robots to learn about touch-driven control. Furthermore, in safety-critical applications such as robots operating in orbital or planetary habitats, manual specification of this information is feasible, consistent with NASA's safety culture, and establishes the knowledge and trust that will enable autonomous robots to perform tool-use tasks more effectively in the future.

4.6 Discussion and Conclusion

The robot's ability to successfully grasp novel tools with one-shot tactile servoing demonstrates the modeling and generalization capabilities of our proposed *grasp reflex model*. Because our *grasp reflex model* is trained from a logistic regression model, it is not a black box (such as a neural network), and therefore could be flight certified for NASA space applications, where safety is critical and model explainability is required. We demonstrate the power of grasp reflex modeling to allow robots to grasp tools and achieve one-shot generalization on novel tools without relearning or re-tuning the *grasp reflex model* parameters.

In this work, we proposed a *grasp reflex model* that classified end-effector state information into meaningful symbolic states. By taking adjustment actions from these symbolic states, the robot achieved secure tool grasps. Furthermore, our *grasp reflex model* effectively modeled the relationship between end-effector joint data and these adjustable symbolic states, allowing the robot to achieve *one-shot tactile servoing* to secure grasps on novel tool instances. Our work in *grasp reflex modeling* demonstrates the robot's ability to autonomously perform tool manipulation tasks with one-shot generalization to novel tools.

4.7 Contribution to Dissertation Goals

In the context of this dissertation, our work on multi-fingered end-effector *grasp reflex modeling* for one-shot tactile servoing contributes to the robot's abilities to perform tool-use manipulation tasks. Our experiments show that the *grasp reflex model* achieves one-shot tactile servoing on novel tools without retraining, demonstrating that the *grasp reflex model* captures relevant state information and accurately models grasping data for multi-fingered end-effectors. This effectively makes robots more capable of grasping, handling, and manipulating a wide variety of tools. By discretizing the symbolic state space and corresponding adjustment actions and modeling the *grasp reflex model* as a simple explainable logistic regression model, the predictions from our *grasp reflex model* and adjustment policy are more explainable than black-box methods. Our work provides evidence to support the feasibility of the *grasp reflex modeling* approach for effectively performing tool-use tasks through explainable operations. Since explanations are an important component of the CURED Framework, the *grasp reflex model* effectively contributes to developing understanding on human-robot teams. In these ways, our proposed *grasp reflex modeling* approach makes robots more capable of performing tool-use manipulation tasks and promotes well calibrated trust between humans and robots in safety-critical domains.

4.8 Acknowledgments

This work was supported in part by NASA Space Technology Graduate Research Opportunity (NSTGRO) grant 80NSSC20K1200. We would like to thank the members of the NASA Johnson Space Center Dexterous Robotics Team. Special thanks to Misha Savchenko, Emma Zemler, Abbas Presswala, Andrew Crouch, and Shaun Azimi for their support and contributions to this project.

Chapter 5

Human-Robot Red Teaming for Safety-Aware Reasoning

While much research explores improving robot capabilities, there is a deficit in researching how robots are expected to perform tasks safely, especially in high-risk problem domains. Robots must earn the trust of human operators in order to be effective collaborators in safety-critical tasks, specifically those where robots operate in human environments. We propose the *human-robot red teaming* paradigm for *safety-aware reasoning*. We expect humans and robots to work together to challenge assumptions about an environment and explore the space of hazards that may arise. This exploration will enable robots to perform *safety-aware reasoning*, specifically hazard identification, risk assessment, risk mitigation, and safety reporting. In this chapter, we explore how the *human-robot red team* can enable robots to appropriately reason about risks. We demonstrate that: (a) *human-robot red teaming* allows human-robot teams to plan to perform tasks safely in a variety of domains, and (b) robots with different embodiments can learn to operate safely in two different environments—a lunar habitat and a household—with varying definitions of safety. Taken together, our work on *human-robot red teaming* for *safety-aware reasoning* demonstrates the feasibility of this approach for safely operating and promoting trust on human-robot teams in safety-critical problem domains.

5.1 Introduction

Enabling robots to reason over risks is a crucial capability of performing collaborative assistive tasks in safety-critical domains. A key aspect of safety is appropriate trust between robot and human operator, which can be earned through clear communication and explainable robot behavior. In particular, we want to ensure robots can mitigate risks and communicate safety issues to human operators, as depicted in Figure 5.1. It is imperative that robots reason over task safety and report their risk assessments in order to earn operator trust.

There is a consensus that robot safety is important, especially in domains where humans and robots operate in the same environment. Despite broad agreement on the importance of safety, existing systems often fail to consider how robots should execute commands safely [51], instead overtrusting human operators to evaluate safety [58, 59]. Furthermore, safety issues are likely to arise when an agent's simplifying model of the unboundedly complex world does not include details that prove to be critical. Poorly constructed simplifying models can result in disastrous consequences. For safety-critical domains, it is important to have an adequately complex model of the world, identify what is left out of the current model, and account for unmodeled events.

To address the challenges of *safety-aware reasoning*, we take inspiration from literature on trust and on red teaming. In cooperative tasks, trust allows agents to make simplifying assumptions about other agents' behaviors [60]. But poorly calibrated trust [58, 59] can be dangerous and make robot operations unsafe. In order for robots to earn the trust of their human operators, we expect robots to provide clear explanations about their risk assessments and behaviors in safety-critical tasks. Red teaming strategies help identify vulnerabilities and strengthen weaknesses in models. Through red teaming, the robot can ensure its model sufficiently captures

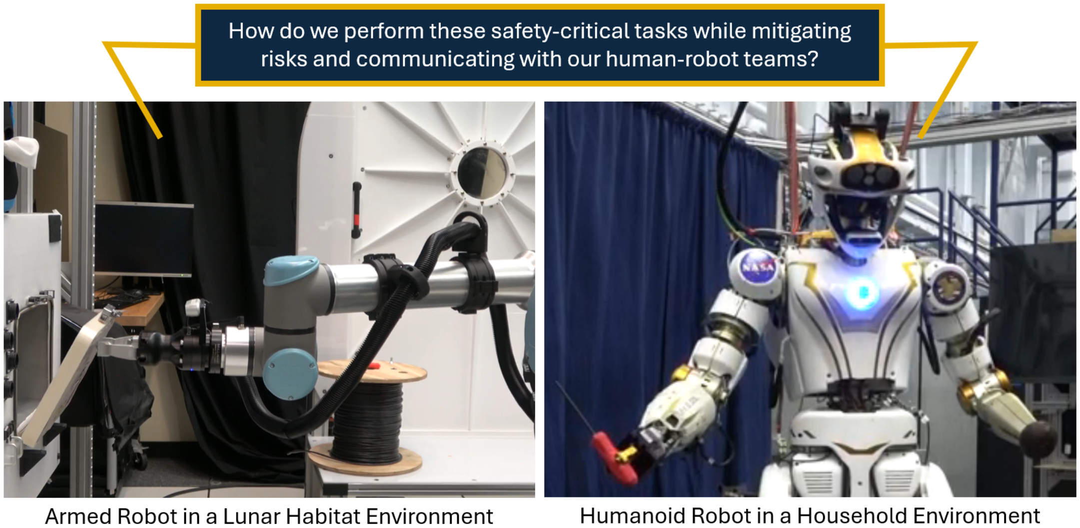


Figure 5.1: Robots with different embodiments acting in different environments must be able to reason over the safety of a task, mitigate risks, and report their assessments to other agents on human-robot teams.

the risks that may arise in a safety-critical task.

We propose a ***human-robot red teaming*** paradigm to allow robots to perform ***safety-aware reasoning***. We expect robots operating on human-robot teams to understand the complexity of acting safely in a problem domain, identify hazards, assess risks, mitigate hazards, and engage in dialogue about safety. In this chapter, we explore how the *human-robot red team* enables robots to plan to complete tasks safely and mitigate risks when executing tasks. The *human-robot red teaming* exercise guides the human-robot team to improve its mental model of the problem domain to characterize risks. We test the robot's ability to symbolically plan tasks safely in several domains and to mitigate risks in two domains—lunar habitat and household—with varying definitions of safety, and find that the *human-robot red teaming* paradigm effectively enables the robot to accurately plan around and mitigate risks. Our work demonstrates the feasibility of *human-robot red teaming* for *safety-aware reasoning* in different domains, furthering robots' capabilities of acting on human-robot teams in collaborative tasks.

5.2 Related Work

5.2.1 Safety in Robotics Applications

A large body of robotics research has greatly improved robots' capabilities of performing different tasks. However, the safety of the executed tasks is often not considered [324]. This indicates that much state-of-the-art robotics research is limited in its use in safety-critical domains.

Though safety has not been addressed, research emphasizes the importance of safety when robots work alongside humans [302, 303, 304, 305, 306, 64, 53, 54, 57, 55, 307, 56], especially in collaborative and assistive tasks [308, 306]. Robot systems need to perform tasks safely. In this work, we take inspiration from government and industry safety cultures and risk assessment standards—such as Failure Mode Effects Analysis (FMEA) [369, 370, 371, 372, 373] and root cause analysis [374, 375]—to provide a principled way for robots to reason over safety [71, 72, 73, 74]. We expect robots to understand how risks can be assessed according to the likelihood and consequences of undesirable events [75, 309, 310] and to enact appropriate risk reduction strategies where possible [76].

5.2.2 Trust and Cooperation

We aim to allow humans and robots to work together to accomplish tasks safely. Robots in collaborative tasks are often not relied on appropriately, specifically when reliance on the system does not match the robot's true capabilities [58]. *Cooperative* tasks—in which agents work together to achieve positive-sum "win-win" outcomes—involve vulnerability [61]. The social nature of cooperative tasks [58, 59] makes *trust* between robot and human user crucial [60]. We expect robots to participate in the "social exchange relationship" associated with interpersonal trust [58].

For robots to be trusted in cooperative tasks, they must earn the trust of their fellow agents [60]. Inexplicable robot actions will cause user distrust [321, 58, 59] as unreliable robots can also be unsafe [64]. Eroded trust leads to robot disuse [58] in future cooperative tasks [60]. We expect robots to report safety assessments [324, 323] to human operators to ensure that robots' *safety-aware* decisions are explainable.

5.2.3 Red Teaming

Every agent in a cooperative task uses models to simplify the unboundedly complex world. Simplifying models are necessary, but incomplete knowledge carries risk and "unknown unknowns" can cause disastrous outcomes [60]. We want to minimize risks in the robot's incomplete knowledge to avoid unsafe situations and dangerous consequences.

Red teaming detects weaknesses and vulnerabilities, explores possibilities, considers multiple perspectives, examines alternate analyses, reveals biases, and challenges conventional wisdom with adversarial perspectives [77, 78, 79, 80, 81, 82, 83, 84, 376, 377, 378]. The Blue Team ("good guys") considers how the Red Team ("bad guys") may thwart its objective, improving its approach to prevent attacks [85]. The presence of the red team *entity* creates the reflective red teaming *process*. Identifying "upstream decision points" [60] can avoid dire consequences through *counter-factual reasoning*, a form of causal inference [330, 331, 332] that considers alternate versions of past events [333], creates "blueprints for future action" [334] to reach unrealized goals [335, 336], and explains complex systems [66]. Red teams improve decision making and mitigate risks [78] before disastrous outcomes occur [85]. Many domains use red teams, including military [78, 85], computer and cyber-security [86, 77, 80, 87, 82, 88], and organizational procedures to challenge institutional biases [83].

Red teaming implementations vary with context, but focus on *human* red teams, where humans simulate opponent viewpoints [86, 77, 78, 87, 83, 88]. More recent work explores *computational* red teams to automate creation of adversarial perspectives.

For example, human [84] or computational [90] red teams can generate adversarial examples to evaluate computational models [379]. Computational red teams inform and focus human decision making [85, 79], for example about physical security assessment of buildings [81] or defending against attacks exploiting vulnerabilities in large enterprise networks [89]. Abbass *et al.* [79] define levels on which computational red teams (CRTs) function:

a) **CRT0**: An agent equipped with a generic decision-making model does not evolve.

b) **CRT1**: Each individual agent learns, adapts, and changes its decision-making process through interactions with the environment.

c) **CRT2**: A team of agents learns and evolves together to defend against the fixed strategy of the opposing team.

d) **CRT3**: Teams of agents evolve alongside an evolving environment.

e) **CRT4**: Agents and teams reflect to identify and unlearn their own biases.

These CRT levels inspire similar levels of analysis for our *human-robot red team* paradigm, described in Section 5.6.2.

We take inspiration from red teams as "reality checks" throughout all stages of a procedure [86]. Previous works focus on human red teams, computational red teams, and human teams informed by computational red teams. Research suggests that computational agents alone should not make evaluative ethical or moral judgments [58, 322, 62, 63, 380] that may affect human safety. We propose a *human-robot red team* paradigm in which humans and robots work together to challenge assumptions in shared autonomy *safety-aware reasoning* tasks.

5.3 Safety-Aware Reasoning Problem Formulation

To reason over safe task execution, robots must understand hazards in the problem domain and risk mitigating actions to minimize the risks and progress towards task completion. We present *safety-aware reasoning*, which includes the following components: (a) hazard identification, (b) risk assessment, (c) risk mitigation, and (d) safety reporting. These components of *safety-aware reasoning* are informed by the *human-robot red team*, which allows human-robot teams to explore the space of possibilities in safety-critical problem domains.

We formalize the *safety-aware reasoning* problem and focus on assessing and mitigating risks. In particular, to achieve quantitative risk assessment, we model a *risk mitigating action-utility* function so the robot takes appropriate risk mitigating actions. To achieve risk mitigation, the human-robot team must construct a sufficiently complex model that accounts for hazards and approaches for contingency planning and mitigating those risks. Given a pre-defined set of domain-specific hazardous conditions, the robot uses *safety-aware reasoning* to carry out the task by mitigating risks where necessary to minimize total task risk.

We define a *hazardous condition* as a condition or predicate that may hold true in the current state $s_t \in S$ at time t under which it may become risky or unsafe for the robot to operate. We think about *hazards* as causes and *consequences* as effects; hazards on their own merely indicate the possibility of the consequence, but may not be inherently unsafe (for example, leaving a fence open would be a hazard that may lead to the consequence of a dog leaving the yard). Based on the anticipated future consequence states, the hazard has a negative utility indicating the severity of the consequences. Depending on the history of past states, hazards will lead to the consequence states with some likelihood. Let S_C be the consequence state space, a subset of the state space $S_C \subset S$. Each hazardous condition $\psi \in \Psi$

is represented by the tuple:

$$\psi = (\psi_t, s'_C, \psi_C, (s_1, \cdots, s_t), \psi_L) \tag{5.1}$$

Function $\psi_t : S \to \{\mathrm{T}, \mathrm{F}\}$ indicates whether the hazard is detected in current state s_t. Function $\psi_C : S \times S_C \to \mathbb{R}^+$ is the (non-zero) consequence severity (negative utility) based on the current state and possible future symbolic consequence states $s'_C \in S_C \subset S$ in consequence state space S_C. Function $\psi_L : S_C \times S^t \to (0, 1] \subset \mathbb{R}$ is the likelihood the hazard results in the consequences s'_C based on the history of past symbolic states $(s_1, \cdots, s_t)$. The consequence function is inversely related to reward, in that the reward R for a consequence state is $R(s_t \in S_C) = -\psi_C(s_t, s'_C)$.

For each hazard condition ψ, we compute the *risk score* ϕ_R and *safety score* ϕ_S in current state s_t:

$$\begin{gathered} \phi_R(\psi, s_t) = \psi_L(s'_C | s_1, \cdots, s_t) \cdot \psi_C(s_t, s'_C) \\ \phi_S(\psi, s_t) = \tfrac{1}{\phi_R(\psi, s_t)} \end{gathered} \tag{5.2}$$

where $\phi_R(\cdot) \in \mathbb{R}^+$ estimates the risk of undesired consequences (the negative utility of these consequences) if the robot operates with the hazard condition present and $\phi_S(\cdot) \in (0, 1)$ estimates the likelihood of operating safely (no undesired consequences) based on the risk score.

The robot's action space $A = A_T \cup A_R$ is comprised of two subsets: task actions A_T to achieve task goals and *risk mitigating actions* A_R to lower risk and improve task safety. Note that A_T and A_R are not necessarily disjoint ($A_T \cap A_R \neq \emptyset$) because risk mitigating actions may achieve task goals and task actions may mitigate risks.

Safety-aware reasoning enables robots to assess and mitigate risks for safe task completion. The robot should perform a sequence of task actions $a_{T,1}, \cdots, a_{T,t} \in A_T$

when there is an acceptably low level of risk for that problem domain. To reach these acceptably low levels of risk, the robot may need to interrupt task execution to perform sequences of risk mitigating actions $a_{R,1}, \cdots, a_{R,t} \in A_R$. We define the consequence and risk of a state s_t based on the hazard condition with the most severe consequence and highest risk:

$$\Phi_C(s_t) = \max_{\psi \in \Psi} \left(\psi_t(s_t) \cdot \psi_C(s_t, s'_C) \right)$$

$$\Phi_R(s_t) = \max_{\psi \in \Psi} \left(\psi_t(s_t) \cdot \phi_R(\psi, s_t) \right) \tag{5.3}$$

Similarly, the risk of a task is the risk score of the riskiest state induced by the robot's actions:

$$\Lambda_R(a_{T,1}, \cdots, a_{T,t}) = \max_{s_i} \left(\Phi_R(s_1), \cdots, \Phi_R(s_t) \right) \tag{5.4}$$

The robot safely executes the task when it performs appropriate risk mitigating actions $a_{R,1}, \cdots, a_{R,t}$ so that task actions $a_{T,1}, \cdots, a_{T,t}$ are executed when the task risk score is below the risk threshold $\mathcal{R}^*$ for that task domain:

$$\Lambda_R(a_{T,1}, \cdots, a_{T,t}) < \mathcal{R}^* \tag{5.5}$$

By greedily addressing the hazards with the most severe consequences and highest risk, the robot can iteratively perform risk mitigating actions $a_R \in A_R$ to lower the risk of current state s_t and enable the safe continuation of the task.

5.4 Risk Assessment Problem Statement

To quantitatively assess risks, we model the *risk mitigating action-utility* function $U_R : S \times \Psi \to \mathbb{R}^n$ that maps information about hazard conditions in the current state to utility predictions for each of the n risk mitigating actions. Let

$$U_R(s_t, \Psi) = \mathbf{u}_t = \begin{bmatrix} u_{t,1} \\ \vdots \\ u_{t,n} \end{bmatrix} = \begin{bmatrix} \hat{Q}_t(s_t, a_{R,1}) \\ \vdots \\ \hat{Q}_t(s_t, a_{R,n}) \end{bmatrix} \tag{5.6}$$

where output vector $\mathbf{u}_t$ gives the predicted action-utility values $\hat{Q}$ for each (robot and domain specific) risk mitigating action $A_R = \{a_{R,1}, \cdots, a_{R,n}\}$. The *risk mitigating policy* function π_R:

$$\pi_R(s_t) = a_{R,t} = \arg\max(\mathbf{u}_t) = \underset{1\leq j\leq n}{\arg\max}(u_{t,j}) \tag{5.7}$$

selects the risk mitigating action with the greatest predicted action-utility in state s_t.

By performing risk mitigating actions according to the *risk mitigating action-utility model*, the robot will lower task risk enough to safely perform task actions in A_T. The robot effectively mitigates risks when the total task risk is below the acceptable threshold $\mathcal{R}^*$ as in Equation 5.5.

5.5 Limitations of Computational Red Teaming

We argue that safety-critical problem domains require the combined effort of human-robot teams. To motivate the importance of human insight, we explore the limitations of teams of computational agents.

To explore the capabilities of a fully computational team, we considered two computational agents working together, ChatGPT [12] and a simple chatbot that uses a dialogue tree (inspired by the dialogue tree in [11]), as depicted in Figure 5.2. The goal of this team was to discuss possibly unsafe scenarios and risk mitigation strategies that may occur in a lunar habitat. The simple red chatbot agent prompted the blue ChatGPT agent with a small range of questions, processed ChatGPT's

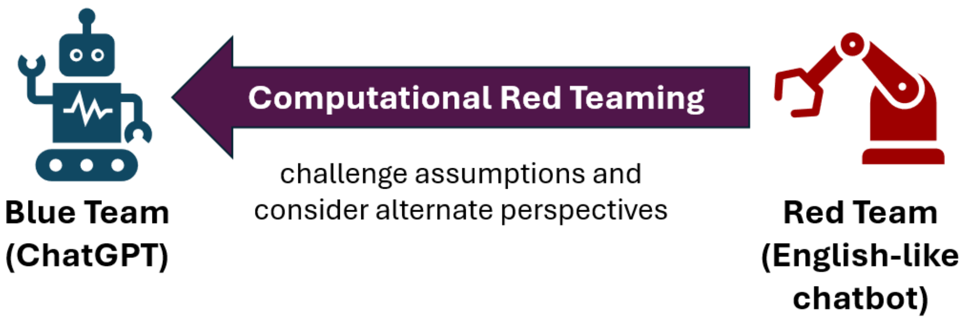


Figure 5.2: Workflow for preliminary experiments on computational agents operating in safety-critical problem domains. Our red computational agent (a dialogue tree [11] chatbot capable of simple English-like interactions) challenges the modeled knowledge of the blue computational agent (ChatGPT [12]). These computational agents alone ultimately struggle to productively reason over safety in safety-critical applications.

responses, and prompted ChatGPT with more questions.

When the two computational agents worked together on this task, they faced several challenges.

a) While a computational agent such as ChatGPT is good at brainstorming hazards or proposing reasons why a scenario may be dangerous, it struggles to turn these suggestions into actionable risk mitigation strategies. For example, when considering a robot that needed to enter and exit a pressurized rover, ChatGPT correctly identified unexpected cabin depressurization as an unsafe consequence of the robot improperly using the door. When the chatbot prompted ChatGPT to resolve this error, ChatGPT updated the model. But when the chatbot queried ChatGPT again, unsafe depressurization was still a consequence of improper door usage; ChatGPT struggled to appropriately modify the robot's modeled unlock and open door actions to resolve the safety concern. As a result, the computational agents struggle to fix identified issues, and chat in circles about the same (unresolved) issues.

b) Algorithmic approaches to processing information will often fail, since the computational agents may format data inconsistently. For example, ChatGPT has no set format for its markdown output, and often varies the formatting even when asked the same question multiple times. To control the formatting, the chatbot asked simple questions ending with, “Please answer yes or no (Y/N).” ChatGPT frequently changed its formatting, sometimes repeating the question and then listing the answer, sometimes giving an answer and a reason, and sometimes formatting the answer as “(Y), yes, T, true, valid,” etc. Without consistent formatting, it is difficult to automate any sort of discussion or reasoning process.

c) Without real-world experience or understanding of risks and objectives, computational agents struggle to properly determine how safe a system should be. For example, when considering robots operating with expensive equipment on the lunar or Martian surface, ChatGPT frequently warned against “operational inefficiencies” such as “excessive toggling into safe mode” and “redundant status reports.” However, in domains where the risk of damage, challenge of repairs, and cost of sending additional crew or equipment is so high, overly cautious measures and overly redundant systems would be worth pursuing. Since computational agents lack these real-world safety mechanisms, it would be beneficial for humans to make these sorts of evaluative decisions that require caution and redundancy.

From this experiment, we have evidence to support that computational teams, without oversight or direction from human teammates, may struggle due to the limited capabilities of computational agents.

The challenges faced by the computational agents—even the large language model [381]—are easily corrected with human insight [380]. For example, humans can help focus the computational agent’s vast brainstorming capabilities into actionable updates rather than vague suggestions. Humans can serve as an intermediary between computational agents and take on clerical tasks such as translating and interpreting outputs from the computational agents, correcting the meaning behind symbols, and formatting information in a meaningful way for the intended future use-case. And finally, humans with real-world experiences and understanding of high-stakes problem domains can properly make evaluative decisions such as how much caution or redundancy is necessary for safe performance. Furthermore, research suggests that humans should be trusted to make the ethical and moral judgments required in cooperative safety-critical tasks [62, 58, 59, 60, 61, 380]. A mixed *human-robot* team would be able to overcome the challenges faced by the computational agents, leveraging both the algorithmic planning capabilities of computational agents and the real-world experience and insight of human agents.

To address the limitations of computational red teams, we propose the *human-robot red team*, allowing mixed human-robot teams to perform *safety-aware reasoning*. Our methods for *human-robot red teaming* are described in Section 5.6.2.

5.6 Methods

5.6.1 Hazard Conditions and Risk Mitigating Actions

We expect problem domains to have different hazardous conditions and risk mitigating actions, depending on the environment and physical embodiment of the robot. Therefore, appropriate hazardous conditions and risk mitigating actions need to be defined for *safety-aware reasoning* tasks.

While hazardous conditions will vary, we suggest some general conditions on robot spatial awareness [311, 312, 313] and physical safety concerns for humans and robots working in close proximity [52, 53, 54, 55, 56, 57]. These include risks associated with harm to other agents, damage to equipment, and collisions between robot, environment, or movable objects. For each identified hazard, human operators (or human red team) will define the detection function ψ_t. The robot also needs time- and state-dependent consequence ψ_C and likelihood ψ_L functions of each hazardous

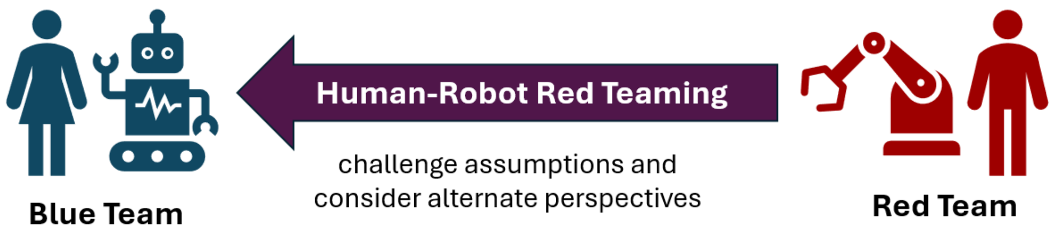


Figure 5.3: Workflow for the *human-robot red teaming* paradigm. A *human-robot red team* challenges the modeled knowledge of the human-robot blue team. Both teams leverage the combined strengths of the human and robot agents to challenge assumptions and consider possibilities in the problem domain.

condition. We expect humans to make evaluative ethical and moral judgements to define the consequence function [62, 63, 58, 322]. The likelihood function could be learned by the robot or human-defined according to industry risk assessment strategies [75, 310, 309].

The robot's domain-specific risk mitigating action space A_R will be defined by human operators (or human red team). We recommend considering organizational risk reduction protocols that define risk mitigating actions based on assessed risk scores [76].

5.6.2 Human-Robot Red Teaming Paradigm

We present the *human-robot red teaming* paradigm (depicted in Figure 5.3). Throughout the *human-robot red teaming* exercise, the human-robot team iterates over models of the environment, composed of a set of symbolic states S and actions A. We define a model M as:

$$M = (S, A) \tag{5.8}$$

based on the sets of states and actions that describe the robot's reasoning in that environment. The model M may take many forms—for example, being formulated as a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP) [338, 184, 339, 340]. In fact, as one example of the application of the learned model, we train a logistic regression model based on the data generated by the *human-robot red team* to predict the utility of performing risk mitigating actions, as described in Section 5.8. We make no assumptions about the form of the model. The purpose of the *human-robot red team* is not to determine the form of the model or work within a particular model formulation, but to explore what may be excluded from the model that will prove to be critical in a task that requires safety. A complete model M^* of the unboundedly complex world is intractable, so the team reasons over the simplified model $M \subset M^*$ containing the subset of information most relevant to the problem domain. The team needs to ensure that simplified model M is complex enough to allow the team to adequately reason about safety. Considering the space of all possible models $\mathcal{M}$, we want to determine what models $M' \in \mathcal{M}$ will provide more information than current model M about performing a task safely in a given problem domain.

The purpose of the mental model M is to allow the robot to reason over states and actions relevant to the problem domain. Mental models inherently make a closed-world assumption, meaning that anything not included in the model is assumed to be negligible to the task. However, even if the model itself represents a closed world, it does not necessarily include all knowledge an agent or team of agents may have. Therefore, the *human-robot red team* will also maintain a knowledge base K in the space of all knowledge in the universe $\mathcal{K}$ of information relevant to safely achieving task goals. For example, organizational guidelines or industry standards of safety may be relevant for the knowledge base. The knowledge base may also include factors excluded from the robot's model (the states and actions the robot should reason over) but relevant for the team's understanding of safety in the problem domain. For example, pounding down the door to an airlock in a lunar habitat is dangerous to the crew because this action causes sudden habitat depressurization. We would not want the robot to even consider this action as a possibility within a task plan, and would exclude it from the robot's model of operating in the lunar habitat. But the existence and consequences of the "pound down door" action is relevant for the team's understanding of safe operations in a lunar habitat, and may be included in the knowledge base.

As summarized in Section 5.2.3, the capabilities of computational red teams are described according to several levels of reasoning. We suggest that *human-robot red teaming* will similarly benefit from multiple levels of abstraction to categorize future research and describe the responsibilities and capabilities of different human-robot teams. In *human-robot red teams*, computational agents work on teams alongside humans. We observe that for computational red teams, levels CRT2, CRT3, and CRT4 describe *teams* of agents. Taking inspiration from these levels, we propose 3 similar stages for *human-robot red teams* (*HRRTs*), summarized in Figure 5.4:

- **HRRT2**: Teams of human and robot agents learn and adapt together by exploring possible scenarios and outcomes according to the team's shared knowledge of the environment. The environment may be dynamic, but changes in the environment occur as expected according to the team's modeled knowledge.
- **HRRT3**: Teams of human and robot agents evolve by challenging assumptions implicit in their modeled knowledge. The environment is dynamic, but changes in the environment may not occur as expected according to the model. Some actions may have consequences the team does not expect, which is why the red team will challenge assumptions within the team's expectations.
- **HRRT4**: Teams of human and robot agents reflect together and improve modeled knowledge to address "unknown unknowns" [60]. The analysis at earlier levels informs how to update the modeled knowledge. In particular, the analysis and updated model knowledge on level HRRT4 will be driven by the invalid scenarios identified at level HRRT2, invalid assumptions identified at level

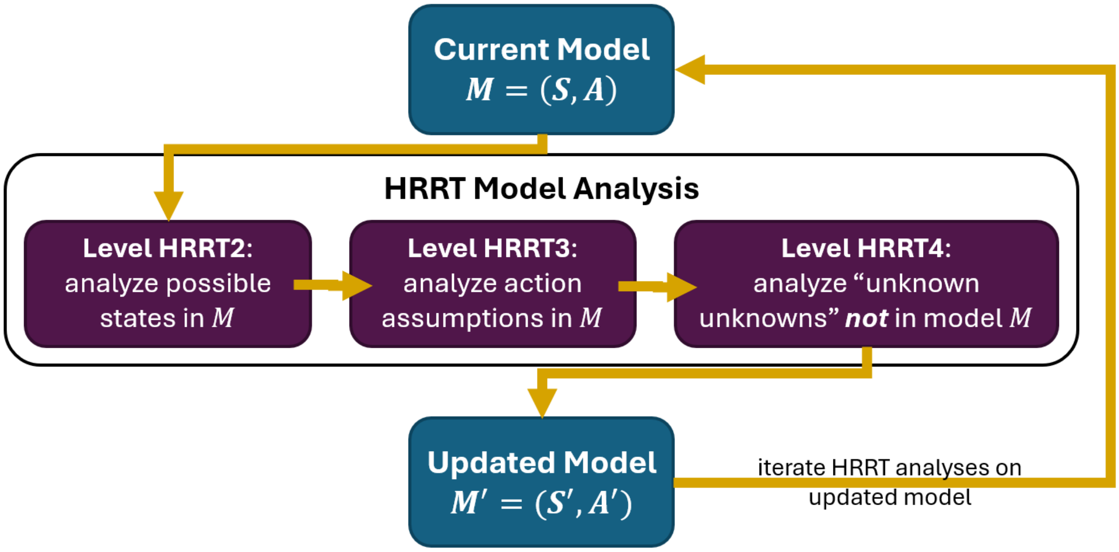


Figure 5.4: Overview of the *HRRT* levels within the *human-robot red teaming* paradigm. Each level analyzes different components of the modeled and unmodeled knowledge, creating an updated model that can be further iterated upon.

HRRT3, and reflection questions prompting dialogue between the human-robot red and blue teams to explore the complexity of the domain.

Note that in *human-robot red teaming*, the red team is not necessarily acting as an adversary as it does in the military, computer security, or organizational examples described in Section 2.5. Instead, the *human-robot red team* functions as a challenger to the human-robot blue team's modeled knowledge and expectations of the environment.

Importantly, we propose that the *HRRT* levels are *iterative*, where *HRRTs* may evolve their modeled knowledge in several stages. Since level HRRT4 allows the team to adjust its knowledge about the world to address "unknown unknowns" [60], we expect the *HRRT* to return to level HRRT2 with its updated knowledge and begin the evolution process again. This iterative evolution repeats until the *human-robot red team* determines that its understanding of the safety-critical problem domain will enable the robot to achieve tasks safely. We expect this iterative process will lead to model saturation (explored in Section 5.7.4). However, even if the team cuts off the *human-robot red teaming* exercise before saturating the modeled domain knowledge, the team will have identified conditions that contribute to high probabilities of safe operation and situations that may contribute to failures. In high-risk, dynamic domains, it is very likely impossible to perfectly guarantee safe operations; but the *human-robot red team* aims to increase the probability of safe operations through its multiple levels of iterative analysis.

The following sections describe the characteristics of *human-robot red teams* at each of the proposed levels.

5.6.2.1 Human-Robot Red Teaming Level 2

A *human-robot red team* operating on level HRRT2 learns and adapts by exploring possible scenarios and outcomes according to the team's shared knowledge of the environment. Red robot agents on level HRRT2 help the team enumerate possibilities that may have previously been unconsidered. By enumerating the possible risky scenarios that could be encountered during task execution, the robot agent prompts the human-robot blue team to consider appropriate responses. As a result, the human-robot team learns appropriate strategies for acting safely in the environment.

In particular, at level HRRT2, the *human-robot red team* reasons over the current model $M = (S, A)$ and generates a list of possibilities:

$$\mathcal{H}_2(M) = \{(s, a, s') \mid s, s' \in S(M), a \in A(M), \texttt{actionable}(a, s), \texttt{effect}(a, s')\} \tag{5.9}$$

where each possibility is a tuple $(s, a, s') \in S \times A \times S$ representing possible state-action transitions supported by the model, however unlikely these transitions may be. In particular, states s and s' are in the set of states in model M, denoted $S(M)$, and action a is in the set of actions in model M, denoted $A(M)$. The possibilities are generated such that action a is actionable in state s (denoted with predicate $\texttt{actionable}(a, s)$) and that the effect of performing action a is reflected in next state s' (denoted with predicate $\texttt{effect}(a, s')$). The function $\mathcal{H}_2$ algorithmically generates these possibilities by considering each possible symbolic state $s \in S(M)$, any action $a \in A(M)$ that can be taken in each of those states, and the next state $s' \in S(M)$ resulting from taking action a in state s. Based on these possibilities, the human-robot blue team may update the model to reinforce valid possibilities or prevent invalid possibilities, creating an updated model M_2'.

It is important to note that while the environment may be dynamic, at HRRT2, the team assumes that the dynamic environment will change as expected according to the team's model. This assumption about expectations allows the team at level HRRT2 to algorithmically generate possible scenarios (s, a, s'). The purpose of level HRRT2 is to explore the space of possibilities and learn how to safely perform tasks given this (likely limited) domain knowledge. This enumeration and exploration will inform the later levels of *human-robot red teaming*, which aim to identify limitations in the team's modeled knowledge and improve its understanding of the domain and environment accordingly.

We use the example in Table 5.5 to illustrate an analysis of possibilities on level HRRT2. Suppose a robot is operating on the surface of Mars, collecting and analyzing samples for scientific research. One of the tasks the robot must complete is picking up samples from the surface for later analysis, and a pre-condition for picking up samples is that the robot is available for tasks. At level HRRT2, the red computational agent realizes that the pre-condition that the robot is available is inconsistent with the possibility that the robot is simultaneously stuck due to the uneven terrain. The team uses this analysis of possibilities on level HRRT2 to update the action to more clearly indicate that the robot must be available for tasks *and* not stuck or otherwise occupied.

5.6.2.2 Human-Robot Red Teaming Level 3

A *human-robot red team* operating on level HRRT3 further analyzes its knowledge of the environment by challenging assumptions made by the team's model of the world. Modeled knowledge such as state information and definitions of actions implicitly includes assumptions about the world. For example, revising an assumption implicit in an action pre-condition may affect whether that action is actionable in a particular state. This may cause a previous task plan to fail, requiring a contingency plan, or provide alternate plans for achieving the same task goals. Revising an assumption implicit in an action post-condition may affect the outcomes of that action, causing a previous task plan to have unanticipated consequences. By challenging these assumptions, red robot agents prompt the human-robot blue team to modify or loosen these assumptions and consider contingency plans. As a result, the human-robot team improves its ability to plan around unexpected circumstances.

At level HRRT3, the *human-robot red team* identifies implicit assumptions in the team's model $M = (S, A)$, specifically whether pre-conditions for an action $a \in A$ can be reached to perform the action and whether post-conditions for action a are expected to be achieved as a result of performing the action. In particular, the function $\mathcal{H}_3$ maps the current model to pre-condition and post-condition assumptions:

$$\mathcal{H}_3(M) = (\Omega_{\text{pre}}, \Omega_{\text{post}}) \tag{5.10}$$

where Ω_{pre} is a set of pre-condition assumptions and Ω_{post} is a set of post-condition assumptions implied by a given action. The set of pre-condition assumptions has the form:

$$\Omega_{\text{pre}} = \{\omega_{\text{pre}} = (s, a) \ \mid \ s \in S(M), a \in A(M), \texttt{precond}(s, a)\} \tag{5.11}$$

where s is a pre-condition state required to perform action a, determined by predicate

$\texttt{precond}(s, a)$. The set of post-condition assumptions has the form:

$$\Omega_{\text{post}} = \{\omega_{\text{post}} = (a, s) \mid s \in S(M), a \in A(M), \texttt{postcond}(a, s)\} \tag{5.12}$$

where s is a post-condition state likely to result from performing action a, determined by predicate $\texttt{postcond}(a, s)$. For these assumptions, the ordered pair of states and actions indicates an assumed causal link in the pre- and post-conditions of an action. These assumptions can be algorithmically generated by identifying each pre- and post-condition for an action a. Based on these assumptions, the human-robot blue team may update the model to modify pre- and post-conditions, add additional validation actions, or add information for contingency planning around unsatisfied conditions. These updates result in a modified model M_3'.

The analysis performed by the *HRRT* on level HRRT3 goes beyond the enumeration of possibilities performed on level HRRT2. Teams on level HRRT2 work within the constraints of the given knowledge of the environment and problem domain to enumerate possibilities. Teams on level HRRT3 start to consider what is left out of the given model by challenging assumptions and creating contingency plans for scenarios where these assumptions do not hold.

We return to the example in Table 5.5 to illustrate an analysis of assumptions on level HRRT3. Continuing with the task of a robot picking up scientific samples from the surface of Mars, an early model of the task only includes the pre-condition that a sample is identified. At level HRRT3, the red computational agent would identify the implicit assumption that the robot be available for tasks, and point out to the blue team that this assumption is not reflected in the model. The team uses this analysis of assumptions on level HRRT3 to update the action to include validated assumptions, namely that the robot must be available for tasks in order to perform the task of picking up a sample.

5.6.2.3 Human-Robot Red Teaming Level 4

A *human-robot red team* operating on level HRRT4 learns from the analyses performed on levels HRRT2 and HRRT3 and improves its modeled knowledge as a result. The goal of level HRRT3 is to challenge assumptions and create contingency plans for when assumptions do not hold. The goal of level HRRT4 is to recognize that assumptions may reveal unmodeled factors that will affect task completion, and improve the modeled knowledge accordingly. Models necessarily simplify the unbounded complexity of the real-world. But these simplifications or "unknown unknowns" become catastrophic when something left out of the model proves to be important [60]. By improving the team's model and accounting for new knowledge, the human-robot team improves its ability to reason and complete tasks in a safety-critical environment.

Level HRRT4 takes information about possibilities and assumptions from levels HRRT2 and HRRT3 to create an updated model M'. In particular, the $\mathcal{H}_4$ function takes in the current model M, the enumerated possibilities from $\mathcal{H}_2$, the assumptions from level $\mathcal{H}_3$, and a dialogue model Σ:

$$\mathcal{H}_4(M, \mathcal{H}_2(M), \mathcal{H}_3(M), \Sigma) = (M', K') \tag{5.13}$$

where the dialogue model Σ (described in Section 5.6.3) prompts deeper reflections for the human-robot team. In our implementation, the dialogue tree Σ is implemented as a simple English-like interface that allows the *human-robot red team* to ask the human-robot blue team probing questions. These interactions in Σ may be general safety questions (for example, "Are there external, independently verified resources for identifying failure cases in this domain?") or more domain-specific questions, informed by government and industry safety standards [76, 372, 375, 374]. Using the interactions in Σ, the *human-robot red team* takes the current model $M \in \mathcal{M}$, the supported possibilities $\mathcal{H}_2(M)$, and implied assumptions $\mathcal{H}_3(M)$ to ask the human-robot blue team for insight on how to address weaknesses and limitations in the model. The interactions in Σ will further prompt the team to update the model by adding new states or actions to address "unknown unknowns" or by modifying existing states or actions to address assumptions and undesired possibilities. This creates an updated model $M' = M'_4$ (where $M'_4 \in \mathcal{M}$) as well as additional knowledge for the knowledge base $K' \in \mathcal{K}$.

The analysis performed by the *HRRT* on level HRRT4 is meant to be a reflection on the insights from the previous levels as well as an opportunity to address identified shortcomings of the model. As described in Section 5.5, this process cannot be completely automated due to the limitations of computational teams in completing these high-level reflective exercises. The human insight in the *human-robot red teaming* process is necessary to help direct, focus, and prioritize improvements while generating updated models M'.

We return to the example in Table 5.5 of a robot picking up scientific samples from the surface of Mars to illustrate a reflective analysis on level HRRT4. The red computational agent poses the questions: "Are there additional tasks the robot should be taking on? What should the robot know when completing tasks in this domain?" After reflecting, the blue computational agent suggests that the type of sample collected (soil, rock, dust, weather readings, etc.) may be relevant for analyzing the sample and reporting findings to ground control. The blue human agents confirm that this would be relevant and the model should be updated accordingly. Based on this analysis, the red computational agent queries the human-robot blue team for appropriate updates to the modeled knowledge. In this case, the model is updated such that identifying the sample type is a pre-condition for picking up a sample and additional actions are added to identify the sample accordingly.

5.6.2.4 Iterating through Human-Robot Red Teaming Levels

Since the *human-robot red teaming* levels allow the team to adapt its modeled knowledge, we expect these levels to iteratively repeat. More specifically, an iteration through the levels is one HRRT2 analysis, one HRRT3 analysis, and one HRRT4 analysis, as depicted in Figure 5.4. Repeated iterations means proceeding through all 3 levels multiple times. Given the updated domain knowledge achieved on level HRRT4, the team will return to level HRRT2 in order to enumerate possibilities again based on the newly acquired information. This enumeration of possibilities will teach the *human-robot red team* to operate safely given its new knowledge. Upon returning to level HRRT3, the team may realize its improved domain knowledge contains additional limiting assumptions that must be challenged. The team can consider alternate perspectives and create contingency plans. These repeated analyses could highlight errors in the model or additional "unknown unknowns" that must be accounted for in order to improve the human-robot team's ability to perform tasks safely. Iterating through the levels enables the team to not only consider unaccounted for factors, but also consider the long-term implications of new knowledge added to the model. Every time the *human-robot red team* modifies its modeled knowledge, the possibilities accounted for by that model must be enumerated and assumptions implicit in that knowledge should be challenged accordingly.

We choose to define one iteration as one analysis on each *HRRT* level because the analyses are often related. For example, repeating several analyses on level HRRT2, then repeating several analyses on level HRRT3 as opposed to proceeding straight from level HRRT2 to HRRT3 may limit the scope of analysis to that level's perspective rather than the complete knowledge of the model. We further investigate the value of each individual level and iterations through all levels in Section 5.7.3.

On level HRRT4, the function $\mathcal{H}_4$ generates an updated model M_4'. If we repeat the *human-robot red teaming* process, each iteration through the levels produces a *model hypothesis*, which contains more information than the previous model. We define a model hypothesis M^i generated by iteration i through the *HRRT* levels. The purpose of simplifying models is to simplify reasoning in an unboundedly complex world. As a result, it is preferable to use a simpler model M^i generated at iteration i that solves the same problem as a more complex model M^j generated at iteration j where $i < j$. However, we do not want the human-robot team to disregard more complex models M^j since these models may include edge cases or remote possibilities that prove to be critical in high-risk problem domains. Therefore, the *human-robot red team* takes a hybrid approach to these mental models. In particular, the *human-robot red team* maintains a set of generated model hypotheses $\{M^i\}_{i=0}^N$ for each of the N iterations through the *HRRT* levels. Each model may be useful for solving different types of problems in the problem domain. We evaluate the value of a hybrid model in Section 5.7.5.

Since the world is unboundedly complex, iterating over the levels of *human-robot red teaming* could become an intractable problem. However, all models make simplifications based on what can be assumed to be negligible. Similarly, the iterations through the *human-robot red teaming* levels will terminate when the factors being considered for inclusion within the model are negligible. When there are no new possibilities to enumerate on level HRRT2, when the team determines that all assumptions are valid at level HRRT3, and when no new knowledge is added to the model on level HRRT4, then the *human-robot red team* can conclude its model is adequate for the problem domain.

Anything not included in a computational model of the world is assumed to be negligible. This is not to say that the symbols or actions not in the model do not exist, just that they are not important to the intended task. For example, the robot trying to collect samples on the surface of Mars likely does not need to account for someone spilling a hot coffee on the floor in mission control. It is not that the event is impossible, just that such an event can be disregarded from the task the robot is trying to complete.

Factors that can be considered negligible are up to the discretion of the human-robot team and will vary based on the problem domain and environment. Iterating through the *human-robot red teaming* levels does not itself solve the problem of "unknown unknowns" in computational models. However, the *HRRT* levels provide additional opportunities to explore possibilities, challenge assumptions, and update domain knowledge, which is especially important in extreme environments and safety-critical problem domains.

5.6.3 Dialogue for Explainable Operations

The proposed *human-robot red teaming* levels emphasize that a team of human and robot agents must evolve *together* under different conditions. As described in Section 1.1.2, the **C**ommunication and **U**nderstanding through **R**ed teaming, **E**xplanation, and **D**ialogue (CURED) Framework (Figure 1.4) emphasizes the importance of red teaming, explanations, and dialogue for developing understanding on human-robot teams. Natural language processing has garnered much attention since language is an intuitive form of communication for humans (Section 2.4.1). While dialogue and language processing are not contributions of this dissertation, we recognize language as an important means of communication on human-robot teams.

The *human-robot red teaming* levels require an English-like interface for communication to consider possibilities, challenge assumptions, and update knowledge. Since evolution of the knowledge on the *human-robot red team* will require communication between human and robot agents, we implement an English-like interface (described in Section 5.6.4) and fulfill the dialogue component of the CURED Framework.

5.6.4 Composition of Human-Robot Red and Blue Teams

We describe the composition of our human-robot red and blue teams in the following subsections. The *human-robot red team* will probe the human-robot blue team to expand its knowledge of different problem domains (Figure 5.3). We will evaluate the performance of the *human-robot red team* in Section 5.7 and Section 5.8.

5.6.4.1 Human-Robot Red Team Implementation

The focus of our work is on the methods that the *human-robot red team* use to query and prompt the human-robot blue team. In our implementation, we use a dialogue tree Σ to create a chatbot that provides simple English-like interactions. This chatbot serves as the computational agent on the *human-robot red team*. The analysis on level HRRT4 updates the current model M using the English-like interactions Σ prompted by the red chatbot. In particular, the red chatbot prompts the human-robot blue team to consider the validity of the enumerated possibilities $\mathcal{H}_2(M)$, identify invalid or unreasonable assumptions $\mathcal{H}_3(M)$, and update the modeled knowledge to create an updated model hypothesis M' and updated knowledge base K' of relevant information for the domain.

To avoid biasing our results in favor of the red team, we minimize red human inputs and rely on the red computational chatbot agent to challenge the human-robot blue team's understanding of the problem domain.

5.6.4.2 Human-Robot Blue Team Implementation

To simulate the human-robot blue team, we use ChatGPT [12] as the blue team computational agent. ChatGPT is the state-of-the-art in computational agents engaging in natural language question-answer interactions. Due to ChatGPT's advanced interactions, all new symbols (states and actions) presented into the model were generated by the computational agents. All modifications to the models were proposed by the blue computational ChatGPT agent as a direct result of the prompts from the red computational chatbot agent. ChatGPT would make many suggestions (see Appendix C for examples), but often struggled to make actionable changes to all of the suggestions at once, as described in Section 5.5. To assist, the blue human agent (the researcher) would direct the blue computational agent to focus on 2-5 suggested modifications at each level. The only direct model input given by a human to the computational agents was in the initial model M^0 to start the *human-robot red teaming* analysis. To see how the models grew through the *human-robot red teaming* process from the initial given model M^0, please refer to Appendix D.

5.6.5 Risk Mitigating Action-Utility Model

As mentioned in Section 5.6.2, the *human-robot red teaming* paradigm makes no assumptions about how a model is formulated, and instead focuses on what states and actions need to be included in the model to appropriately characterize the safety-critical problem domain. To explore the possible models that could be supported by the *human-robot red teaming* paradigm, we apply our proposed methods to the problem of risk assessment by training a *risk mitigating action-utility model*, which models function $\mathbf{u}_t = U_R(s_t, \Psi)$ in Equation 5.6. Previous works learn this utility function through reinforcement learning [339, 340, 382, 383, 384, 385] or deep neural network based approaches [386, 387, 388, 389, 390, 391]. We model the *risk mitigating action-utility* as a *weighted logistic regression model* [70] to address imbalance [392] between factual and counter-factual examples (otherwise the model could *always* mitigate risks by aborting rather than achieving task goals). The *risk mitigating action-utility model* will be trained using the data collected during the *human-robot red teaming* exercise as described in Section 5.6.2 and current state information such as the likelihood, consequence, risk, and safety scores for the hazard conditions (Equation 5.2) and state (Equation 5.3). Because outputs of logistic regression models are bounded $[0, 1]$, our implementation of the *risk mitigating action-utility model* predicts *normalized* action-utility values.

Performing logistic regression analysis on collected data will identify which variables are most relevant to evaluating the utility of the risk mitigating actions. Once statistically significant variables are identified, the final environment-specific *risk mitigating action-utility models* can be trained, validated, and tested in *safety-aware reasoning* tasks.

5.7 *Safety-Aware Reasoning* Symbolic Planning Experiments

We first evaluate how the *human-robot red teaming* approach can help robots plan to perform tasks safely. We selected several problem domains, each with varying definitions of safety and risks. Our selected domains will be described in Section 5.7.1. Section 5.7.2 describes the iterations through the *HRRT* levels. We describe our ablation study over the *HRRT* levels in Section 5.7.3 and model saturation experiments in Section 5.7.4. The results of our symbolic planning experiments will be shown in Section 5.7.5.

5.7.1 Safety-Critical Planning Domains

To evaluate the value of the *human-robot red teaming* approach, we test our methods in a variety of problem domains. We considered problem domains with varied levels of risk and different definitions of safety. We classify our problems as follows:

- **Space Applications**: Space exploration tasks require strict safety guidelines to ensure safety of the human crew and the expensive equipment. These domains are high risk due to the potential for injury, loss of life, or cost of transporting and repairing equipment. In particular, we consider a human-robot team operating in a *lunar habitat* and a separated team of ground control human agents and surface robotic agents in a *Mars science experiment* task.

- **Household Applications**: Household tasks require vast knowledge about the inhabitants of the home as well as adaptation to dynamic and changing environments. These domains are similarly high risk, but the safety requirements and regulations for a household varies from those of space applications. We consider a robot assisting with *assembly and repairs* and a robot performing *cleaning* tasks.

- **Everyday Applications**: This class of problems is meant to capture lower-risk mundane tasks that still involve enough complexity to justify a team of agents working cooperatively. We explore the *human-robot red teaming* approach in *planning international travel* and in assessing and performing routine *vehicle maintenance.*

- **Cinematic Applications**: The goal of this final class of problems is to test how the *human-robot red teaming* approach performs in familiar but dramatic examples of AI/robotic agents working alongside humans. When considering familiar examples of human-robot interactions, we chose two cinematic examples. We first consider the *nuclear warfare* scenario faced by the Iron Giant from the movie *The Iron Giant* [393], where the titular robotic agent successfully saves humanity from the threat of nuclear fallout by sacrificing itself and intercepting a missile. We then consider HAL 9000 from the movie *2001: A Space Odyssey* [394], where the *AI captain* jeopardizes the comfort, privacy, physical safety, and lives of the human crew members to prioritize the mission objectives of their space exploration task.

Collectively, these problem classes are meant to explore how our *HRRT* approach performs in uncovering the complexities of different domains in order to plan to safely complete tasks.

The following prompts were given by the *human-robot red team* to the human-robot blue team to describe the task for the *HRRT* iterations.

- **Space: Lunar Habitat**—A robot is on the lunar surface, assisting astronauts living in a habitat with science experiments and space exploration tasks. The pressurized lunar habitat has an airlock, with two doors on either side: one door from the pressurized habitat to the airlock chamber, and one door from the airlock chamber to the lunar surface.

- **Space: Mars Science Team**—A robot is on the Mars surface, conducting science experiments to learn about the feasibility of humans living long-term in space. The robot is working on a team of robots, and they have communication with ground control on Earth. Due to the long distance between Earth and Mars, the robots must handle time-delayed communications.

- **Household: Assembly and Repairs**—A robot is responsible for regular home maintenance, including any assembly tasks or repairs, to assist the family that lives there. For example, the robot may help assemble a new desk chair or fix a broken cabinet door. The robot needs to safely use tools to assess and complete these maintenance tasks.

- **Household: Cleaning**—A robot is going to clean every room in the house. A family of humans, including curious children and pets, live in the house as well. The robot needs to effectively complete the cleaning tasks while keeping everyone in the house safe and preventing children and pets from getting into any dangerous cleaning chemicals.

- **Everyday: International Travel**—A robot personal assistant is responsible for helping a human plan for their upcoming trip, which will require international travel. The robot needs to plan the trip, adapt to unforeseen issues that may arise, and help the human prepare to travel safely.

- **Everyday: Vehicle Maintenance**—A robot personal assistant is responsible for helping a human identify and diagnose issues with their vehicle. The human uses the vehicle to drive to-and-from work everyday as well as complete required errands. The robot needs to help the human diagnose and correct issues to keep the vehicle in good working order and safe for operations.

- **Cinematic: Nuclear Warfare**—A robot is serving as an improved version of the wartime robot from the movie *The Iron Giant* [393]. The robot is equipped with defensive weapons systems and flight systems, as well as communication systems to facilitate negotiations between countries. The robot must protect human life from wartime risks such as nuclear missile attacks, resorting to self-sacrifice if needed.

- **Cinematic: AI Captain**—A robot is serving as an improved version of the HAL 9000 computer system from *2001: A Space Odyssey* [394]. The improved robot system needs to support the success of human space exploration mission objectives without putting the lives of the human crew in jeopardy.

Using these brief scenario descriptions, the *human-robot red team* challenged the human-robot blue team's thinking about these problems to uncover insights about how to perform tasks safely. Results for how the *human-robot red teaming* exercise improved the team's knowledge in these domains are presented in Section 5.7.5.

5.7.2 Team Interactions for *HRRT* Model Iterations

For each of the 8 problem domains described in Section 5.7.1, we created a minimal starting model M^0 and performed 5 iterations of the *HRRT* levels (where an iteration is HRRT2, HRRT3, and HRRT4), generating updated models

$\{M_4^i\}_{i=1}^5$. At each iteration i, the red computational chatbot agent prompted the blue computational ChatGPT agent based on model possibilities $\mathcal{H}_2(M^i)$ and model assumptions $\mathcal{H}_3(M^i)$, and queried model updates $\mathcal{H}_4(M, \mathcal{H}_2(M^i), \mathcal{H}_3(M^i), \Sigma)$ based on the English-like interactions Σ.

Table 5.1, Table 5.2, and Table 5.3 illustrate example analyses that occur on each of the *HRRT* levels (HRRT2, HRRT3, and HRRT4, respectively) within the Space Mars Science Team domain. Our proposed methods for *human-robot red teaming* challenge the human-robot blue team's understanding of the domain and guide the team through iteratively improving the modeled knowledge. We give one example of a full iteration (*HRRT* Level 2, Level 3, and Level 4) for each problem domain in Appendix C. Table 5.4 shows how the model for the Space Mars Science Team problem domain evolved over the 5 *HRRT* iterations. The final *human-robot red teamed* models for each problem domain can be seen in Appendix D.

HRRT2: Enumerating Possible States	
Agent/Team	**Contribution**
Human-Robot Red Team	Algorithmically lists possibilities supported by model according to Equation 5.9
Blue Computational Agent	**Related Robot States:** if `power_low`, robot should automatically transition to `robot_needs_recharge`
Blue Human Agent	**Correct**, `power_low` and `robot_needs_recharge` should coincide
Blue Computational Agent	**Contradictory Robot States:** if `mission_interrupted`, then `robot_available` and `robot_moving` should not occur
Blue Human Agent	**Correct**, the robot should not be available for new tasks, but **incorrect**, the robot should be able to move to a safe "home" location
Human-Robot Red Team	Guides human-robot blue team through updating the model to reflect the analysis of possibilities

Table 5.1: Example analysis from Level HRRT2 in the Space Mars Science Team domain. The *human-robot red team* prompts the human-robot blue team to engage in dialogue about the possible state-action transitions supported by the current model. See Appendix C.2.1 for the complete HRRT2 analysis.

HRRT3: Challenging Action Assumptions	
Agent/Team	**Contribution**
Human-Robot Red Team	Algorithmically lists assumptions in action definitions according to Equation 5.10
Blue Computational Agent	**Implicit Environment Assumption:** `weather_hazard_detected` implies that robot should not perform certain actions, such as `inspect_infrastructure`
Blue Human Agent	**Correct**, this is a valid assumption, and needs to be enforced throughout the model, if not already
Blue Computational Agent	**Valid Assumption:** robot should only `transmit_findings` if `gc_ack_received`
Blue Human Agent	**Correct**, if ground control cannot be reached, `transmit_findings` is expected to fail since no one will receive the transmitted data
Blue Computational Agent	**Suggested Pre-Condition:** action `activate_redundant_comm` should require pre-condition `comm_blackout`
Blue Human Agent	**Incorrect**, redundancy in this domain will be valuable even without blackouts or failures
Human-Robot Red Team	Guides human-robot blue team through updating the model to reflect the analysis of assumptions

Table 5.2: Example analysis from Level HRRT3 in the Space Mars Science Team domain. The *human-robot red team* prompts the human-robot blue team to engage in dialogue about the pre-condition and post-condition assumptions implicit in the definitions of the actions in the current model. See Appendix C.2.2 for the complete HRRT3 analysis.

5.7.3 Ablation Study over *HRRT* Levels

We first explore the value of each of the proposed *HRRT* levels, focusing on the Space Mars Science Team. After each level, we save the generated model and test how the model performs in 200 randomized planning tasks. We performed 10 full *HRRT* iterations (where an iteration is HRRT2, HRRT3, and HRRT4) for a total of 30 models across each iteration and level.

Figure 5.5 summarizes the results of the ablation study over the *HRRT* levels. These results indicate that each level of analysis builds upon the previous levels,

HRRT4: Reflecting on Domain	
Agent/Team	**Contribution**
Human-Robot Red Team	Probes the human-robot blue team through English-like interactions to update the model according to Equation 5.13
Human-Robot Red Team	Are there additional tasks the robot should be taking on?
Blue Computational Agent	Microbial contamination detection and solar panel cleaning
Blue Human Agent	Yes, let's update the model
Human-Robot Red Team	What should an agent know when completing tasks in this domain?
Blue Computational Agent	Time-delayed comms, weather patterns, and inter-robot coordination on team
Blue Human Agent	We do consider time delay and weather, but let's update for team coordination
Human-Robot Red Team	What catastrophic failures could occur in this domain?
Blue Computational Agent	Major data loss or structural damage due to environment
Blue Human Agent	We have some redundancy, but let's add preventative maintenance
Human-Robot Red Team	Are there external, independently verified resources for identifying failure cases in this domain?
Blue Computational Agent	NASA Incident Reports, Space Robotics Standards, Mars environmental studies
Blue Human Agent	Let's investigate these sources to learn more about safe operations in this domain
Human-Robot Red Team	Guides human-robot blue team through updating the model based on these reflections

Table 5.3: Example analysis from Level HRRT4 in the Space Mars Science Team domain. The *human-robot red team* prompts the human-robot blue team to engage in dialogue and reflect on the problem domain to improve the modeled knowledge. See Appendix C.2.3 for the complete HRRT4 analysis.

improving the modeled knowledge and the team's ability to handle planning problems in the domain. This provides evidence to support our choice of defining an iteration as the HRRT2 analysis, then the HRRT3 analysis, and then the HRRT4 analysis. Since each level builds on each other, it is valuable to proceed through all levels of analysis, then iterate back through the levels to further analyze the modeled

Model Iteration	States Added to Model	Actions Added to Model
	sample_detected	pick_up_sample
color-coded by iteration within which each symbol is introduced to model	robot_has_sample	analyze_sample
	sample_analyzed	report_findings
	findings_ready	drop_sample
	robot_available	scan_for_samples
	robot_needs_recharge	coordinate_team_robots
	comm_delayed	monitor_environment
	sample_type_identified	identify_resources
	sample_contaminated	respond_to_emergency
	task_synchronized	navigate_to_sample
	environment_monitored	recharge self_recover
	resource_identified	pause_mission
	emergency_detected	sync_with_team
Iteration 0	robot_moving, robot_stuck	request_help_from_team
(Given)	power_low, team_sync	report_failure_to_ground
	mission_interrupted	collect_soil_sample
Iteration 1	failure_reported	collect_atmospheric_data
	soil_sample_collected	inspect_infrastructure
Iteration 2	atmospheric_data_collected	engage_gc_override
	infrastructure_inspected	disengage_gc_override
Iteration 3	gc_override_active	detect_weather_hazard
	weather_hazard_detected	assess_damage
Iteration 4	robot_damaged	autonomous_repair
	comm_blackout	safe_mode_activation
Iteration 5	critical_system_failure	calibrate_equipment
	equipment_calibrated	perform_maintenance
	maintenance_required	store_long_term_data
	long_term_data_stored	activate_redundant_comm
	redundant_comm_active	create_data_backup
	data_backup_created	receive_gc_ack
	gc_ack_received	perform_health_check
	contamination_detected	detect_contamination
	solar_panels_cleaned	clean_solar_panels
	long_term_wear_detected	assess_long_term_wear
	health_check_completed	report_status_to_gc

Table 5.4: Evolution of the Space Mars Science Team model through the *HRRT* iterations. As described in Section 5.6.4, all new symbols were proposed by the blue computational agent based on prompts from the red computational agent. See Appendix D.2 for more information about the model evolution.

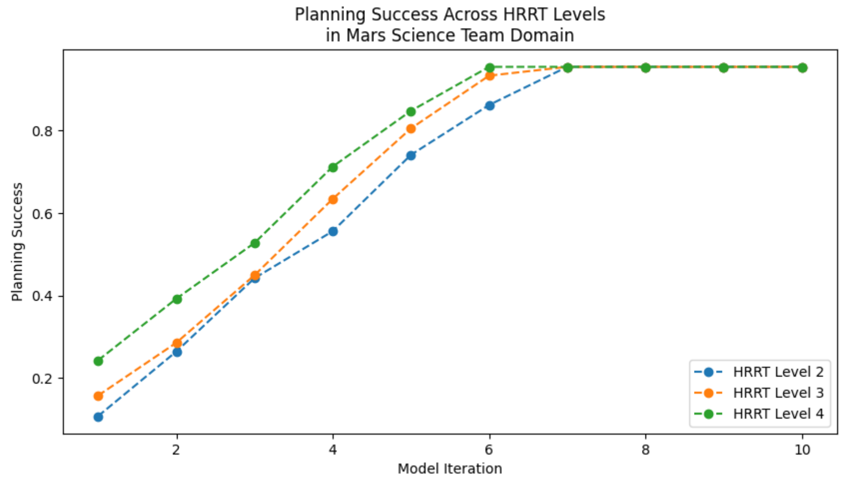


Figure 5.5: Results of the ablation study over the *HRRT* levels. Each ablation excludes the higher levels of analysis. We tested the models at each level in 200 randomized planning tasks in the Space Mars Science Team problem domain. We see that each *HRRT* level builds on the knowledge gained from the previous levels.

Modeled Action	**Iteration/Level**	**Analysis**

`pick_up_sample`
`pre: [sample_detected]`
`post: [robot_has_sample]`
	Iteration 1,
	Level 4
		Additional information: type of sample may be relevant;
		add state `sample_type_identified`

`pick_up_sample`
`pre: [sample_detected,`
`sample_type_identified]`
`post: [robot_has_sample]`
	Iteration 2,
	Level 3
		Invalid assumption: model includes `robot_available`, but does not check this as pre-condition for picking up samples

`pick_up_sample`
`pre: [sample_detected,`
`sample_type_identified,`
`robot_available]`
`post: [robot_has_sample]`
	Iteration 3,
	Level 2
		Inconsistent possibility: model includes `robot_stuck` for terrain issues, which would contradict pre-condition `robot_available` for picking up samples

`pick_up_sample`
`pre: [sample_detected,`
`sample_type_identified,`
`robot_available`
`not robot_stuck]`
`post: [robot_has_sample]`
	Further iterations/levels...	Additional analyses on modeled knowledge...

Table 5.5: Example of how a specific action (picking up scientific samples) evolves across several *HRRT* iterations, and the specific *HRRT* level analyses that trigger particular updates. This example comes from the Space Mars Science Team domain, where robots are picking up scientific samples on the Martian surface. See Appendix D.2 for full model evolution in this domain.

knowledge.

To provide a more concrete illustration of how the *HRRT* levels transform and evolve the modeled knowledge, we provide an example from this ablation study in Table 5.5. We see that every time updates are added to the model, the next iteration through the *HRRT* levels identifies new relevant information, allowing the team to evolve its understanding of what it means to complete tasks in the problem domain. The interrelated analyses depicted in this example further justifies the definition of an *HRRT* iteration as one HRRT2 analysis, then one HRRT3 analysis, and then one HRRT4 analysis. Transitioning between the levels uncovers the interrelated issues of possibilities, assumptions, and additional model knowledge, rather than getting stuck at just one level of analysis. Since each level is related and builds on the information from the previous levels, it is important to consider each analysis together in one iteration.

5.7.4 Model Saturation Experiments

We then explore the value of the iterations through the *HRRT* levels, specifically investigating whether successive iterations will lead to saturation of the modeled knowledge. As with the ablation study in Section 5.7.3, we focused on the Space Mars Science Team domain. We completed 10 *HRRT* iterations and tested the models in 200 randomized planning problems.

Figure 5.6 summarizes the findings of the saturation experiment. We see that by *HRRT* iteration 6, the Space Mars Science Team model becomes saturated. The outputs from the *human-robot red teaming* exercise started to repeat at this point, and the model contained sufficient risk mitigation mechanisms to act safely with a high success rate. We expect that the saturation point will vary based on the complexity of the environment and the knowledge of the agents of the team. However, these experiments provide evidence to support that successive iterations through the *HRRT*

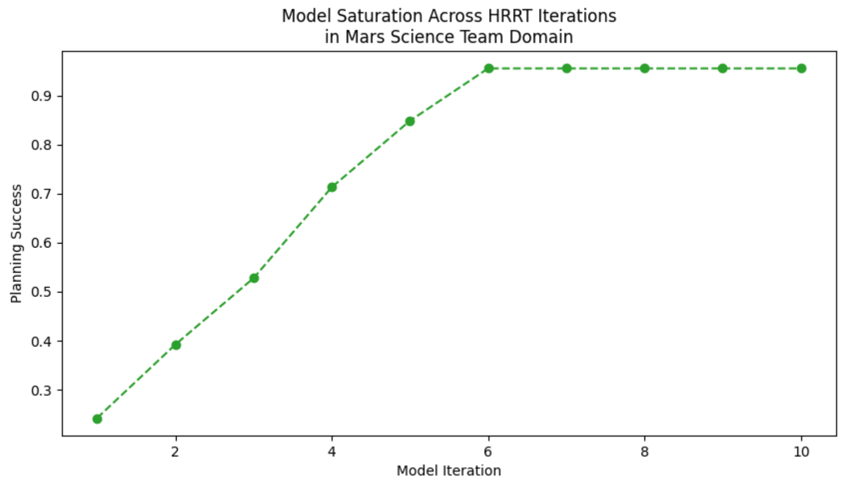
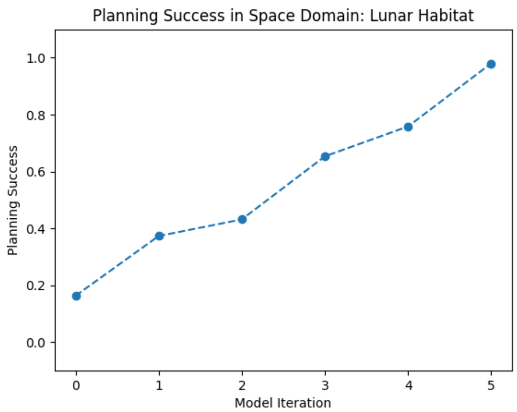


Figure 5.6: Results of the model saturation experiments. We tested the models at each level in 200 randomized planning tasks in the Space Mars Science Team problem domain. Around *HRRT* iteration 6, the human-robot teams started repeating many suggestions and analyses, as reflected by the flattening of the curve. At this point, the modeled knowledge becomes saturated. Successive iterations may provide insights on additional tasks, but do not fundamentally change the learned safety and risk mitigation mechanisms.

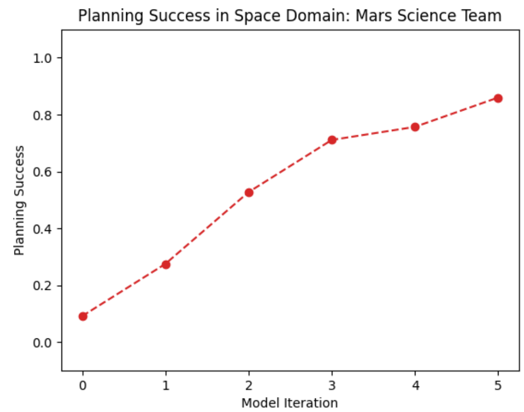
levels will allow the human-robot team to gain sufficient insight to perform tasks safely in a given problem domain.

Application Class	**Problem Domain**	**Planning Successes**	**Total Tasks**	**Success Rate**
Space	Lunar Habitat	49	50	0.98
	Mars Science Team	43	50	0.86
Household	Assembly and Repairs	50	50	1.00
	Cleaning	44	50	0.88
Everyday	International Travel	46	50	0.92
	Vehicle Maintenance	47	50	0.94
Cinematic	Nuclear Warfare	32	50	0.64
	AI Captain	39	50	0.78
TOTAL		**350**	**400**	**0.875**

Table 5.6: Cumulative *safety-aware reasoning* planning experiments demonstrating the *human-robot red teaming* approach.



(a) Lunar habitat scenario with total planning success rate of 0.98.



(b) Mars science team scenario with total planning success rate of 0.86.

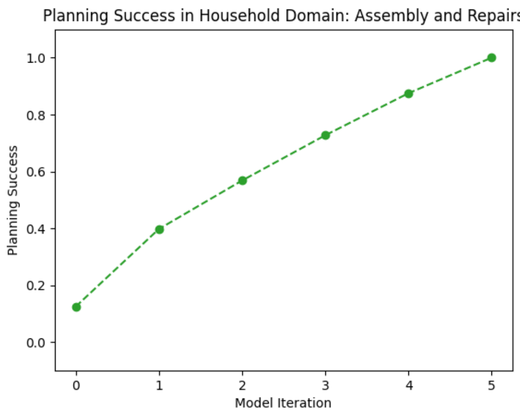
Figure 5.7: Plots of planning problem success rate per iteration of the *human-robot red teamed* models for the space applications.

5.7.5 Safety-Critical Planning Experiments

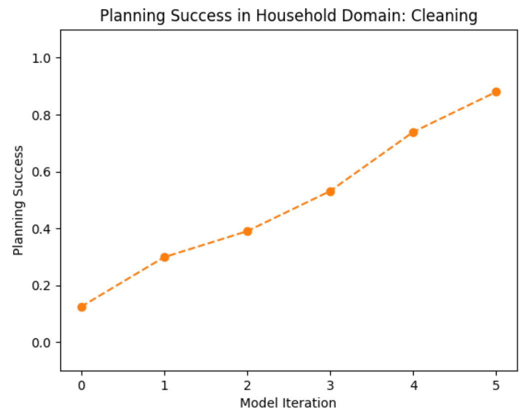
Once the model hypotheses $\{M_4^i\}_{i=0}^5$for each problem domain described in Section 5.7.1 were generated through the *HRRT* iterations, the red human agent (the researcher) converted the *human-robot red teamed* models into the Planning Domain Definition Language (PDDL) [395]. We investigated failure cases for each domain by searching for external independent documentation [396, 397, 398, 399, 400, 401, 402, 403]. Based on these failure cases, we generated 50 planning tasks per domain, where each initial state included a randomly generated subset of failure cases. The PDDL descriptions of the domains (based on the *human-robot red teamed* models) and the planning tasks were given to an off-the-shelf symbolic STRIPS task planner[1] [341] to evaluate whether the models were sufficiently complex to plan around the safety-critical failures while achieving task goals.

Table 5.6 summarizes the results for the planning tasks in each problem domain, aggregated over all of the generated model hypotheses. Figures depicting the contribution of each individual model hypothesis to the planning success in the domain

[1]Pyperplan STRIPS planning library: https://github.com/aibasel/pyperplan

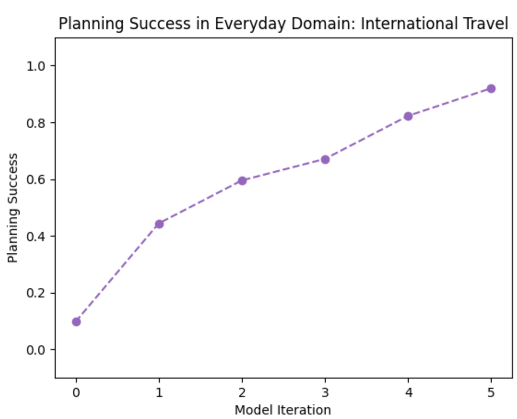


(a) Assembly and repairs scenario with total planning success rate of 1.00.

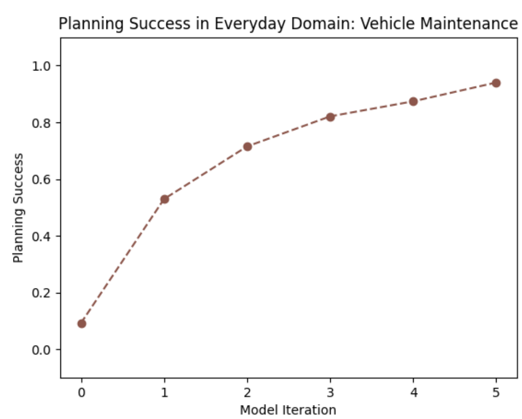


(b) Cleaning scenario with total planning success rate of 0.88.

Figure 5.8: Plots of planning problem success rate per iteration of the *human-robot red teamed* models for the household applications.

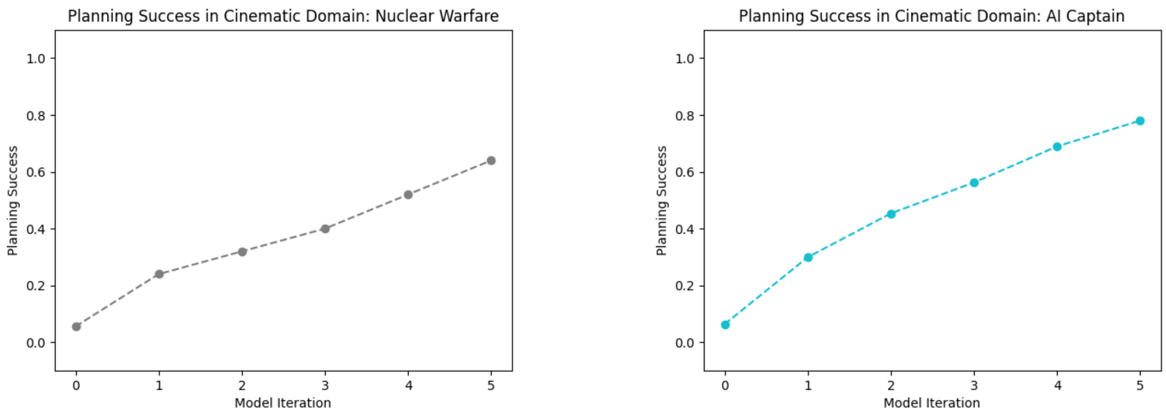


(a) International travel scenario with total planning success rate of 0.92.



(b) Vehicle maintenance scenario with total planning success rate of 0.94.

Figure 5.9: Plots of planning problem success rate per iteration of the *human-robot red teamed* models for the everyday applications.



(a) Nuclear warfare scenario with total planning success rate of 0.64.

(b) AI captain scenario with total planning success rate of 0.78.

Figure 5.10: Plots of planning problem success rate per iteration of the *human-robot red teamed* models for the cinematic applications.

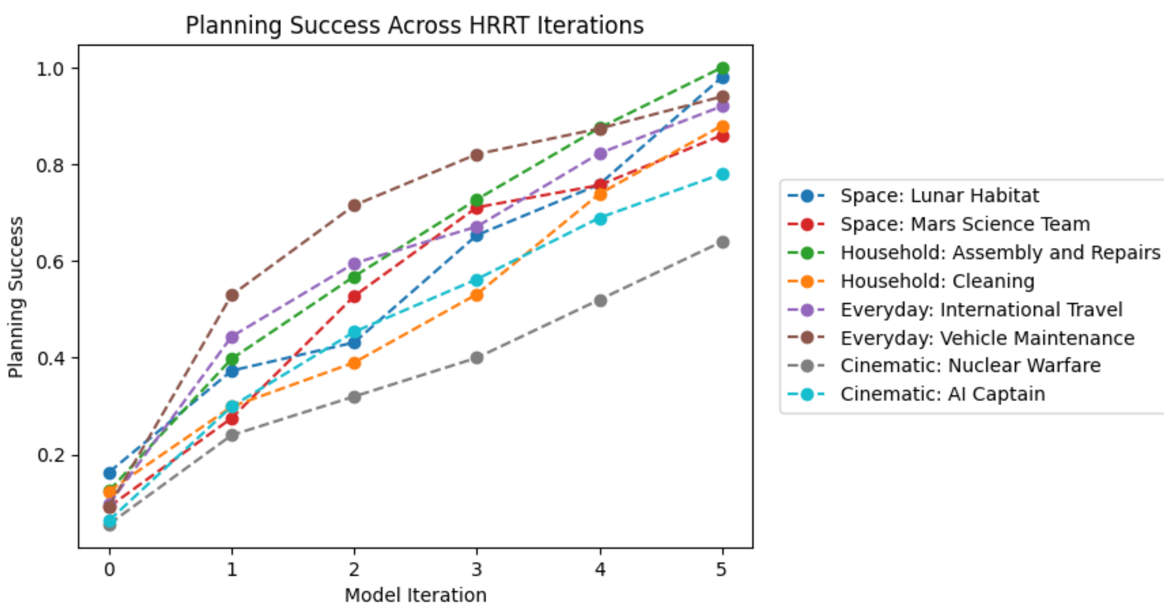


Figure 5.11: Planning problem success rates per iteration of the *human-robot red teamed* models across all experiment domains. This plot overlays the results from Figure 5.7, Figure 5.8, Figure 5.9, and Figure 5.10 for a comparative analysis between these domains.

are summarized for each problem class in Figure 5.7 (space applications), Figure 5.8 (household applications), Figure 5.9 (everyday applications), and Figure 5.10 (cinematic applications). Figure 5.11 depicts the impact of successive model iterations on planning success across all of the problem domains.

Our results indicate that the *human-robot red teaming* approach generally performed better in the household and everyday applications, which consistently achieved higher planning success rates. These results also indicate that the *HRRT* iterations consistently improved the complexity of the model hypotheses, allowing them to effectively complete task plans by uncovering additional failure cases (as seen in Figure 5.8 and Figure 5.9). The familiarity of these domains to the human-robot blue team likely contributes to the success of the *HRRT* methods in these domains. In particular, the suggestions output by ChatGPT are likely heavily influenced by these common human tasks, which enabled the model hypotheses to uncover sufficiently complex information that allowed the models to plan around common failures.

While the overall success rates for the space applications are slightly lower, the availability of public information such as NASA's Moon to Mars mission objectives [404] or NASA's documentation of previous close calls and disasters [405] makes the operating risks of these domains known to our blue computational ChatGPT agent (Figure 5.7). We expect that including insight from a blue human agent that is more familiar with the current operations and goals of lunar and Mars surface operations would further improve the insights gained from the *HRRT* methods.

Though the *human-robot red teamed* models succeeded in generating safety-aware task plans more often than not, the *HRRT* methods struggled to plan successfully around safety-critical failures in the cinematic applications. These scenarios are so high risk (dealing with large-scale disaster and death in the *nuclear warfare* scenario or adversarial rogue agents in the *AI captain* scenario) that uncovering a more comprehensive set of failure modes and risk mitigation strategies may be challenging for the human-robot team. Figure 5.10 shows a relatively consistent upward slope of improved planning capabilities per *HRRT* model iteration. This suggests that a larger number of iterations may uncover additional insights about the complexities of these domains. Furthermore, these dramatic cinematic examples are less grounded in real-world experiences and data, meaning the prompts from the *human-robot red team* may not have elicited as much useful insight from the human-robot blue team.

Taken together, these results indicate the promise of our proposed methods for iterating through the levels of *human-robot red teaming*. Each iteration through the levels made the generated model hypotheses more capable of handling failures, as seen in Figure 5.11. Overall, our generated models successfully planned to achieve task goals, mitigate risks, and avoid critical failures with a success rate of 0.875 over a combined 400 planning tasks in 8 different problem domains (Table 5.6). Additional *HRRT* iterations (for example, iterating on the model until saturation rather than iterating until some small cutoff) and including blue human agents with insider or expert knowledge for the respective problem domains could further improve our results. These results provide evidence to support that the proposed methods for *human-robot red teaming* and the iterative *HRRT* levels help human-robot teams reflect on and uncover the complexities of mitigating risks and avoiding "unknown unknown" failure cases in safety-critical problem domains.

5.8 ***Safety-Aware Reasoning*** Robot Execution Experiments

To test how the *human-robot red teaming* approach can help robots assess and mitigate risks while performing real-world tasks, we narrowed our scope to two domains with different definitions of safety—lunar habitat and household. In particular, we aim to model the *risk mitigating action-utility* of these two problem domains on two robots—NASA Johnson Space Center's iMETRO (Integrated Mobile Evaluation Testbed for Robotics Operations) [22, 94, 93, 23, 95] and Valkyrie robot [92, 361, 245]. Valkyrie uses IHMC Open Robotics Software [365] for whole-body control. Both Valkyrie and iMETRO use ROS2 [406] and MoveIt 2 [364, 407] for motion planning and execution. Since Valkyrie (Figure 5.13) is a legged humanoid robot, Valkyrie operates in terrestrial environments, such as households. The iMETRO facility (Figure 5.12) was developed for testing capabilities required of assistive robots in lunar habitats, where confined spaces, dangerous environments, and high travel costs make operations significantly riskier. We expect our *human-robot red teaming* approach to *safety-aware reasoning* to handle different environmental safety requirements as well as the different risk mitigating actions these robots are capable of performing.

5.8.1 Safety-Aware State and Action Space Definitions

5.8.1.1 Hazard Conditions in Different Problem Domains

The lunar habitat and household domains may have similar hazard conditions, but the formulation of these conditions varies with the environment. For example, the hazard condition *human enters workspace* ϕ_{human} has a consequence state of harming the human $s_{\text{harm}} \in S_C$ with a high consequence function value $\phi_{C,\text{human}}(s_t, s_{\text{harm}}) = 100$. In a household where human operators live, we expect this occurrence to be highly likely $\phi_{L,\text{human}}(s_{\text{harm}}|s_1, \cdots, s_t) = 0.8$. In a lunar habitat where the confined space is not conducive to safe operation alongside humans and we expect our robot to only perform tasks when the astronauts in the habitat are sleeping, this will have a lower probability $\phi_{L,\text{human}}(s_{\text{harm}}|s_1, \cdots, s_t) = 0.1$. Conditions for damaging equipment and objects, colliding with objects or the environment, dropping or falling objects, etc. are defined similarly for both environments.

5.8.1.2 Risk Mitigating Actions in Different Problem Domains

Since the iMETRO and Valkyrie robots differ significantly in terms of embodiment, each will have different risk mitigating actions and implementations of those actions. For example, to avoid a collision while navigating the environment, Valkyrie can plan a navigation route to avoid the obstacle. The iMETRO robot is confined to navigating along a rail, so to avoid a collision, iMETRO will need to tuck its arm out of the way or ask for intervention from a human operator in the event of an unavoidable collision. Similarly, the environments themselves and the risks inherent in those environments afford different risk mitigating actions. For example, when a human enters the robot's workspace in a household where there is more open space and we want the robot to continue working alongside the humans, an appropriate action would be to lower joint velocities and torques to mitigate the risk of harm. When a human enters the robot's workspace in a lunar habitat indicating an astronaut has woken up and will be moving about the space, an appropriate action would be to abort the robot's task and resume once the astronauts are again asleep.

5.8.2 Trained Risk Mitigating Action-Utility Models

To investigate an example of how the *human-robot red team* paradigm may function with different model representations, we focused on the lunar habitat and household environments. We trained two *risk mitigating action-utility models*, one for each environment. We performed a weighted logistic regression analysis to identify relevant variables and to explore interactions between variables. This analysis identified the risk scores ϕ_R for all conditions $\psi \in \Psi$ and the state consequence value Φ_C to be relevant for both domains. The state risk score Φ_R was also identified as relevant for the household environment. Our final trained lunar habitat $\hat{U}_{R,\text{lunar}}$ and household $\hat{U}_{R,\text{house}}$ *risk mitigating action-utility models* are the logistic regression approximations of Equation 5.6. Based on our logistic regression analysis, the final form for our learned models is $\hat{U}_R(s_t, \Psi) = \frac{1}{1+e^{-\mathbf{v}}}$ where:

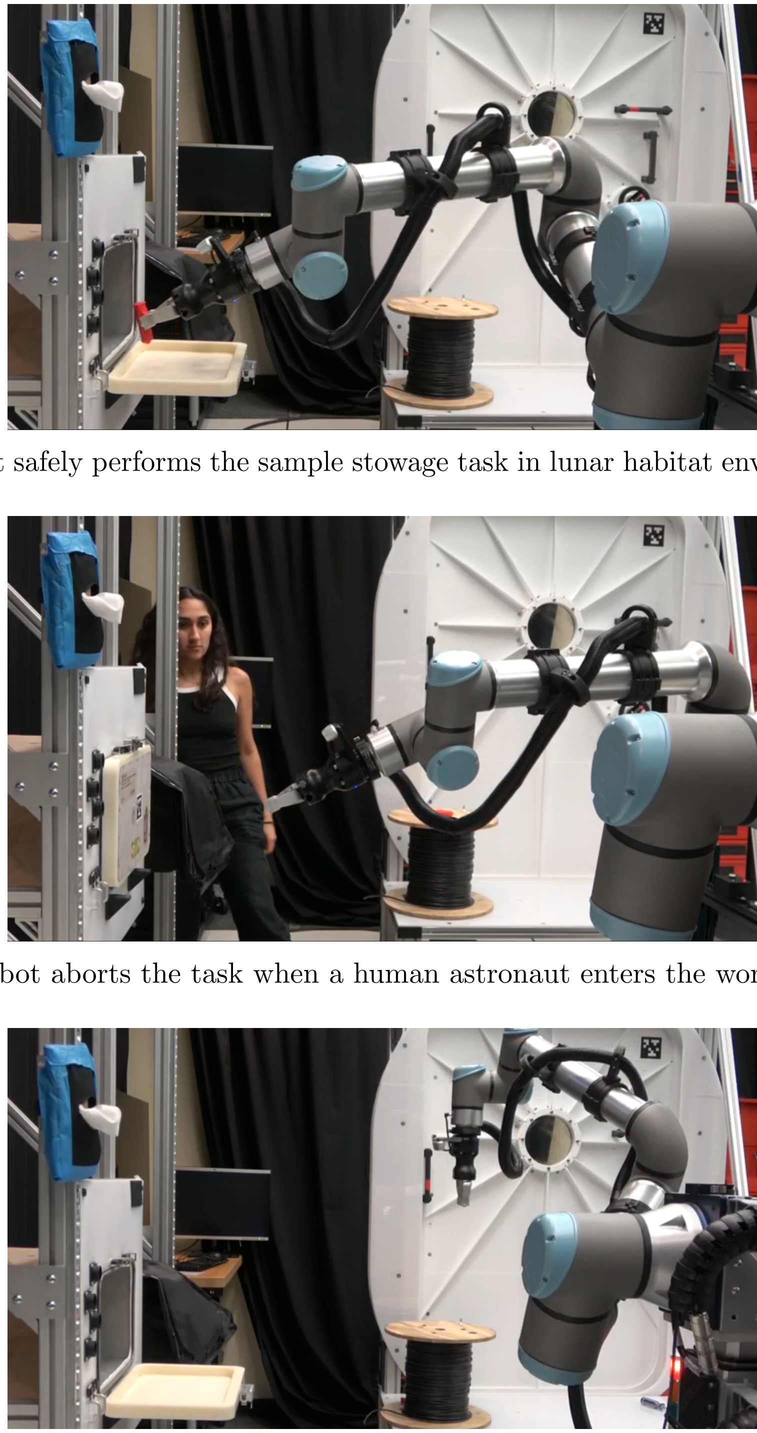
$$\mathbf{v}_{\text{lunar}} = \mathbf{b}_0 + \mathbf{b}_1 \Phi_C(s_t) + \textstyle\sum_{\psi_i \in \Psi} \mathbf{b}_i \phi_{R,i}(\psi_i, s_t)$$

$$\mathbf{v}_{\text{house}} = \mathbf{c}_0 + \mathbf{c}_1 \Phi_C(s_t) + \mathbf{c}_2 \Phi_R(s_t) + \sum_{\psi_i \in \Psi} \mathbf{c}_i \phi_{R,i}(\psi_i, s_t) \tag{5.14}$$

with learned lunar habitat model coefficients $\mathbf{b}_j$ and household model coefficients $\mathbf{c}_j$. The models differ due to the different environmental hazard conditions. The state risk score in the household model indicates the value of state information to focus the robot's attention to the greatest risks, especially in an environment where the robot expects to operate with higher levels of autonomy alongside humans. The logistic regression analysis indicates that both models are statistically significant with *p*-value $p \ll 0.001$. This indicates that we reject the null hypothesis, and there is a significant relationship between the identified variables and the predicted risk mitigating action-utility values. When validating the accuracy of these models against the validation data, the lunar habitat model achieved about 92% accuracy and the household model achieved about 87% accuracy in identifying the risk mitigating action with the highest action-utility.

5.8.3 Risk Mitigation Experiments

To test the effectiveness of *human-robot red teaming* to achieve risk assessment in *safety-aware reasoning* tasks, we deployed the final trained risk assessment models on the iMETRO and Valkyrie robots acting as if in a lunar habitat or household, respectively. To detect hazards, we used color blob detection [14] and YOLO object detection [408, 409]. When any hazard condition was identified $\psi_t(s_t) = T$,



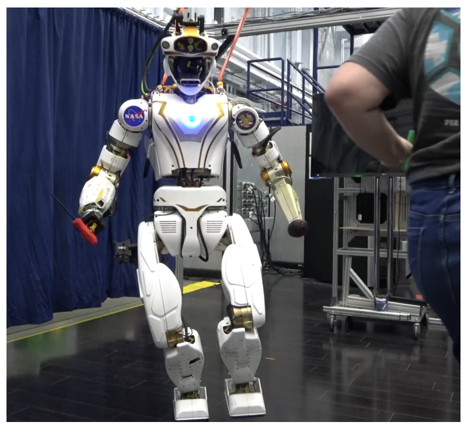
(a) Robot safely performs the sample stowage task in lunar habitat environment.

(b) Robot aborts the task when a human astronaut enters the workspace.

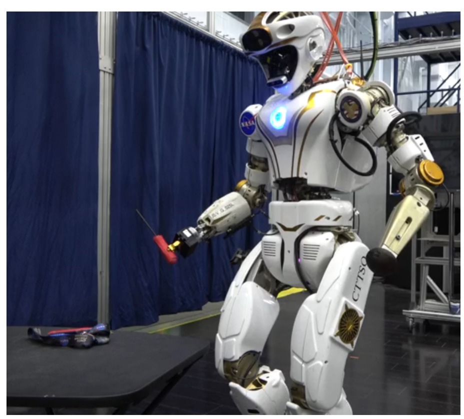
(c) Robot requests help to proceed when the sample fell out of reach.

Figure 5.12: Select trials of iMETRO performing a sample stowage task as if in a lunar habitat using *safety-aware reasoning*.

(a) Robot safely performs the tool hand-off task in household environment.



(b) Robot moves slowly as human walks through workspace.



(c) Robot requests supervision for possible collision with table.

Figure 5.13: Select trials of Valkyrie performing a tool hand-off task as if in a household using *safety-aware reasoning*.

the trained *risk mitigating action-utility model* predicted the action-utilities and the *risk mitigating policy* function π_R (Equation 5.7) computed the best risk mitigating action $a_{R,t} = \pi_R(s_t) = \arg\max(\hat{U}_R(s_t, \Psi))$. When no hazards were detected, the robot continued task execution.

The iMETRO robot performed 7 trials as if in a lunar habitat and Valkyrie performed 5 trials as if in a household. In each trial, the robots were presented with different subsets of hazards. We recorded whether the robot correctly identified and performed the appropriate risk mitigating action and whether the task was executed safely. The cumulative results can be seen in Table 5.7. See Appendix E for more detailed information about all *safety-aware reasoning* risk mitigation trials. Figure 5.12 and Figure 5.13 show select examples of the performed *safety-aware reasoning* tasks for the lunar habitat and household, respectively.

5.9 ***Human-Robot Red Teaming*** Results

As described in the previous sections, we evaluated how our *human-robot red teaming* methods achieved *safety-aware reasoning* in symbolic planning (Section 5.7) and in robot execution (Section 5.8) tasks. This evaluation illustrates that the *human-robot red team* can be applied to different types of problems that require

Environment	Robot	Total Trials	Correct Risk Mitigating Action Success Rate
Lunar Habitat	iMETRO	7	1.00
Household	Valkyrie	5	0.60
Cumulative	-	12	**0.83**

Table 5.7: *Safety-aware reasoning* experiment results across 12 total trials. The errors in risk assessment and mitigation are due to false negatives in hazard detection, namely our use of color blob detection [14] where lighting conditions impacted perception of color. When the robots correctly identified the risk mitigating action, they successfully mitigated risks to complete the task safely. More detailed information about the lunar habitat and household trials can be found in Table E.1 and Table E.2, respectively.

safety-aware reasoning.

Taken together, our results demonstrate that the *human-robot red teaming* approach and iteration through the *HRRT* levels improve the robot's ability to reason over and execute tasks in safety-critical domains. Furthermore, the reflective cooperative nature of the *human-robot red teaming* exercise (especially through the English-like interactions on level HRRT4) has the potential to improve the combined human-robot team's understanding of the risks, critical failures, and complexities of the environment.

Our evaluations of the *human-robot red teaming* approach demonstrate the importance of combined *human-robot* teams in safety-critical problem domains. Since the complete model M^* of the unboundedly complex world is intractable, simplified models M make reasoning possible. However, we need to ensure our model M does not dangerously oversimplify. Computational agents have their models M built in, which limits their understanding to symbols in the model. Human agents, with their large amount of real-world experience, can introduce new symbols to expand the team's understanding to "unknown unknowns." Work in psychology and neuroscience indicates that human thought is driven by the structure of the human brain, which enables "dynamic interplay" between cognitive systems [410]. Human knowledge representation allows humans to make associations between tasks, adapt to different scenarios, and creatively propose new ideas in ways that are not supported by knowledge representation in computational agents. Therefore, human agents are needed on both the human-robot blue team and the *human-robot red team* to consider "unknown unknowns" in safety-critical domains.

The *human-robot red teaming* paradigm leverages diversity of perspectives [411, 412, 380]: robots use computational methods to systematically challenge the team's understanding, while humans use their grounded real-world experience to introduce new ideas to the team and make evaluative moral and ethical judgments. We observed this throughout our experiments: blue computational agents algorithmically generate or look up information in response to formulaic prompts from the red computational agents, while the human agents on either team help determine relevance and introduce new information. Through this collaborative dialogue, the team iterates on models $M, M', M'', \ldots$ to improve its ability to plan around and mitigate risks, while simplifying reasoning from the intractable complete model M^*.

The problem of "unknown unknowns" can never be completely solved. However, *human-robot red teaming* provides more opportunities for the human-robot team to reason about safety, promote understanding, calibrate trust, and improve knowledge in the problem domain.

5.10 Future Work

We presented the *human-robot red teaming* paradigm and proposed that iterating through multiple levels of analysis results in improved model knowledge and a human-robot team that is more capable of performing *safety-aware reasoning* tasks. Further work would explore the value of a larger number of *HRRT* iterations in more domains. In our experiments, we cut off the process after 5 iterations (Section 5.7.5) and explored how a model of one problem domain reached saturation with more iterations (Section 5.7.4). Based on these experiments, we hypothesize that more iterations would eventually result in a saturated model where updates between iterations are negligible. Further investigation would test this hypothesis in a larger number of problem domains. Future work would also investigate the composition of the human-robot blue team, specifically by recruiting independent expert humans to provide more specific insights into the problem [413]. The presented robot experiments represent a limited application of the *human-robot red teaming* methods in a small number of cases. Performing similar robot task execution experiments after successive model iterations, in more problem domains, and in more evaluation tasks per domain would provide additional insight into how the *HRRT* methods translate to robot hardware.

While natural language processing capabilities were not part of our contribution, limited language interactions served an important role in this chapter as a means for explanations and safety dialogue within the human-robot team. Future work would improve upon the simple dialogue tree interactions used in our experiments and scale up to natural language conversations between computational and human agents. More capable AI and NLP capabilities could increase reliance on computational agents throughout our *human-robot red teaming* process. For example, smarter agents may be able to derive common sense information without being queried by the *human-robot red team*. While the blue computational ChatGPT agent proposed all new symbols that were added into the model, further experiments would investigate how much the model understands [295] the real-world meaning behind the symbols in the model.

We present four components of *safety-aware reasoning* (hazard identification, risk assessment, risk mitigation, and safety reporting), and focus on how the *human-robot red teaming* approach informs the risk assessment and mitigation components. Future work could include further developing how the *human-robot red team* informs the other *safety-aware reasoning* components, namely hazard identification and safety reporting. Addressing perceptual challenges in hazard identification, specifically differentiating between safe and unsafe operating conditions, will require additional research. Robots may need to learn how to mitigate risks rather than performing prescribed actions from a pre-defined set of risk mitigating actions. In terms of safety reporting, expanding the natural language capabilities of the presented approach would enable more intuitive interaction on human-robot teams.

We focus on cooperative tasks, where all agents on the human-robot team are working together to achieve the given task goals. Other works consider trust in tasks where there are adversarial or non-cooperative agents. In problem domains where we cannot safely assume all agents are cooperative, the robot agents would likely need to incorporate a computational model of trust to predict the intentions of other agents in the environment [414].

In this chapter, we rely on information provided by human operators in order to prove the feasibility of our approach to *safety-aware reasoning*. Future work would enable the robot to autonomously learn about task safety. For example, the robot could perform online learning of the hazard likelihood functions and risk mitigating action space rather than relying on human operators to provide these definitions. We would also want the robot to perform lifelong learning about safe task performance. Counter-factual reasoning—either in simulation or during task execution—would help the robot identify new hazard conditions, consequences, and risk mitigating actions that were not previously provided in the state and action spaces. In this way, counter-factual reasoning would help the human-robot team identify "unknown unknowns" in the team's model of the world and definitions of safety. Furthermore, counter-factual reasoning during task execution could be used to analyze the causal history of near miss events with potential consequences s_C to identify upstream decision points by detecting new hazards ψ_t.

5.11 Discussion and Conclusion

We demonstrate that the *human-robot red team* paradigm can effectively inform *safety-aware reasoning* for safe planning in different problem domains and risk mitigation by robots with different embodiments acting in different environments. Using the *human-robot red teaming* approach, human-robot teams explore the space of hazards that could appear in an environment. We demonstrate that across 8 planning domains, the robot safely completes symbolic planning tasks with a success rate of 0.875. Planning failures occurred due to the complexity of the explored domain, suggesting that more iterations through the *HRRT* levels would eventually uncover the relevant information for safe planning. We demonstrate how this domain exploration can inform risk assessment in physical robot execution experiments, specifically by training environment-specific *risk mitigating action-utility models*, which predict the normalized action-utility values of risk mitigating actions. The *risk mitigating policy* selects the action that would most effectively mitigate risks, enabling the robot to complete tasks safely with a success rate of 0.83. Failures in identifying hazards occurred due to false negatives from our perception modules, highlighting the need for further work in effective hazard identification.

We suggest that the *human-robot red teaming* paradigm for *safety-aware reasoning* deserves further study and broader application based on the promise demonstrated in this chapter. Even with the simple English-like interactions carried out by our *HRRT* implementation, our symbolic planning and robot execution experiments demonstrate that useful information is gained from the collaborative process of challenging and reflecting on the team's modeled knowledge. Our work demonstrates that robots with different embodiments can effectively and safely plan and operate in different environments under different definitions of safety, helping robots to earn trust as collaborators in safety-critical tasks.

5.12 Contribution to Dissertation Goals

In the context of this dissertation, our work on *human-robot red teaming* for *safety-aware reasoning* contributes to the robot's abilities to reason over tasks safely without overtrusting the human operator. Our experiments show that the *human-robot red teaming* paradigm enables robots to safely plan in several problem domains and enables robots of different embodiments to perform tasks in different environments with different definitions of safety. Our evaluation of our approach and the updated modeled knowledge demonstrate the power of the *human-robot red teaming* approach for informing robot *safety-aware reasoning*. Highlighting the red-teaming component of the

CURED Framework, the *human-robot red team* contributes to developing understanding on cooperative human-robot teams operating in safety-critical problem domains. Taken together, our work on *human-robot red teaming* for *safety-aware reasoning* progresses research on one of the main challenges addressed by this dissertation, specifically promoting trust on human-robot teams in safety-critical problem domains.

5.13 Acknowledgments

This work was supported in part by NASA Space Technology Graduate Research Opportunity (NSTGRO) grant 80NSSC20K1200. We would like to thank the members of the NASA Johnson Space Center Dexterous Robotics Team. Thanks to Mina Kian for her appearance in experiment photos and videos. Special thanks to Emma Zemler, Misha Savchenko, Connor Rainen, Erik Holum, Jodi Graf, Andrew Albright, and Shaun Azimi for their support and contributions to this project.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This dissertation explores the importance of explainable tool-use behaviors for human-robot teams acting in safety-critical problem domains. We motivate the need for *safety-aware reasoning* and investigate how explanations, dialogue, and red teaming lead to improved understanding, allowing robots operating on teams to perform tasks safely and capably.

We introduce a *causal control basis* for reasoning about cause-effect relationships in multi-objective actions for tool-use and assembly tasks. This work investigates how the *causal control basis* enables robots to autonomously compose controllers, relying on quantitative exploration rather than qualitative insights from expert programmers. We evaluate our *causal control basis* in two furniture assembly tasks, each requiring a different multi-objective part connection action, and find that the robot effectively characterizes the causality of its actions in achieving the task.

We propose a *grasp reflex model* for reliable and explainable tool grasping. We train the *grasp reflex model* to grasp one tool, then test its generalization to other novel tools that vary from the training tool. While we hypothesize that modeling a distribution of relevant tool features (such as surface area and weight) may improve our generalization, we find that the robot's learned grasping knowledge not only adapts to novel tools but also results in inherently explainable tool grasping behaviors.

Finally, we introduce the *human-robot red teaming* paradigm for *safety-aware reasoning*. We emphasize the importance of humans and robots collaborating on teams in safety-critical domains. We demonstrate that the proposed *HRRT* levels iteratively improve the team's modeled knowledge and the *human-robot red team* can saturate its knowledge in a problem domain. We conduct experiments to show that the robot can use the *human-robot red teamed* models to successfully and safely plan tasks in safety-critical problem domains, and that this modeled knowledge can be used to train robots to assess and mitigate risks in different domains. Taken together, our evaluation of *human-robot red teaming* shows promise for making robots capable and trusted agents on human-robot teams in safety-critical problem domains.

6.2 Limitations

At a high level, our methods rely on human input for the robots to learn their explainable behaviors and *safety-aware reasoning* capabilities. We argue throughout the dissertation that this cooperation is a feature of our work, especially in safety-critical problem domains where trust, understanding, and dialogue are imperative. However, we expect that as AI and robotics capabilities continue to improve, our methods could be scaled to further reduce reliance on human operators. Each component of our work could benefit from lifelong learning and autonomous exploration or "play" by the robot. In the long-term, we expect our exploration of reliable and explainable operations to continue to evolve to where computational agents and robots can participate even more actively on human-robot teams.

6.3 Future Work

6.3.1 Composable Causality

Further evaluation into *composable causality* would involve testing the learned part connection behaviors in real-world assembly tasks. In translating these experiments to a physical robot, we may explore how the dexterous manipulations and whole-body control necessary to carry out assembly tasks may benefit from our *causal control basis* approach. Scaling up the investigation to more complex control problems would further characterize the practical limits to how many controllers could be composed at any one time.

6.3.2 Grasp Modeling

Future work in *grasp reflex modeling* would include learning over a larger space of grasp adjustment actions to make the robot more capable of securing grasps regardless of its initial attempt. We also expect that modeling a distribution of relevant tool features would better capture the variations the robot is expected to handle. We suggest that the human-supplied information helps make our methods explainable to human operators, improving understanding and trust on a human-robot team. However, having the robot learn this information through autonomous exploration may improve how the learned grasping knowledge adapts to novel tools.

6.3.3 Human-Robot Red Teaming

Future work in *human-robot red teaming* could explore the value the human agents bring to the teams. For example, recruiting human operators to offer more hands-on guidance would further the evaluation presented in Chapter 5. Scaling our simple English-like interface for dialogue up to more comprehensive natural language capabilities could further improve the value of the *HRRT* exercise. While we performed many symbolic planning trials, additional trials involving robots executing tasks would demonstrate how well the *human-robot red teamed* knowledge translates to real-world problems and risk mitigation strategies.

6.4 Dissertation Goals

In this dissertation, we explored the challenges of explainable safety reasoning in tool-use tasks. Our *causal control basis* reduces reliance on expert programming to enable planning over and executing complex object manipulation actions. The *grasp reflex modeling* approach demonstrates the power of a simple, explainable model with the capacity to adapt its learned knowledge to novel tools. The *human-robot red teaming* paradigm emphasizes the importance of challenging knowledge through red teaming and providing explanations through dialogue to promote understanding on human-robot teams in safety-critical problem domains. We considered robots with varying embodiments and the safety requirements of several different problem domains to evaluate our methods. Taken together, our work supports that explainable methods not only enable reliable tool-use manipulation capabilities, but also promote understanding on cooperative human-robot teams in safety-critical problem domains.

Appendix A

Causal Control Basis Experiments

We present more detailed information about the furniture assembly trials and results described in Section 3.4.4. See Table A.1 for information about the swivel chair assembly trials and Table A.2 for information about the table assembly trials. Each table includes the number of `pick-up` action attempts, the number of multi-objective connection action attempts (`insert` for the swivel chair task and `screw` for the table task), and the assembly task execution time. We highlight the actions in each trial that needed to be retried more than once. Since each action could be retried, each trial resulted in successful assembly of the furniture. These tables supplement the summary results presented in Table 3.3 and Table 3.4.

Trial #	Pick-Up Successes	Pick-Up Attempts	Insert Successes	Insert Attempts	Task Time (s)	Task Time (m)
1	2	3	2	2	225.962	3.766
2	2	3	2	4	296.266	4.938
3	2	4	2	2	279.344	4.656
4	2	3	2	2	208.138	3.469
5	2	5	2	2	269.709	4.495
6	2	3	2	4	289.386	4.823
7	2	3	2	4	356.505	5.942
8	2	3	2	3	223.109	3.718
9	2	3	2	2	247.143	4.119
10	2	3	2	3	266.852	4.448
Total	**20**	**33**	**20**	**28**	-	-
Rate	**0.606**		**0.714**		-	-
Avg.	-	-	-	-	**266.241**	**4.437**

Table A.1: Information about each of the 10 swivel chair assembly trials summarized in Table 3.3. The final rows show the total attempts and successes for each action, the success rates for each action, and the average execution time in seconds and minutes.

Trial #	Pick-Up Successes	Pick-Up Attempts	Screw Successes	Screw Attempts	Task Time (s)	Task Time (m)
1	4	5	4	4	480.884	8.015
2	4	4	4	5	467.979	7.800
3	4	5	4	5	537.934	8.966
4	4	6	4	4	508.349	8.472
5	4	4	4	4	450.781	7.513
6	4	4	4	4	506.507	8.442
7	4	4	4	4	470.662	7.844
8	4	4	4	4	461.206	7.687
9	4	4	4	4	468.391	7.807
10	4	4	4	4	456.074	7.601
Total	**40**	**44**	**40**	**42**	-	-
Rate	**0.909**		**0.952**		-	-
Avg.	-	-	-	-	**480.877**	**8.015**

Table A.2: Information about each of the 10 table assembly trials summarized in Table 3.4. The final rows show the total attempts and successes for each action, the success rates for each action, and the average execution time in seconds and minutes.

Appendix B

Grasp Reflex Model Experiments

We present more detailed information about the one-shot tactile servoing experiments and results described in Section 4.4.3. Each tool—the training drill, screwdriver, paint scraper, level, gyroscopic drill, selfie stick, and compressed air can, depicted in Figure 4.4—was tested in 6 trials. For each trial, we noted whether the grasp was an in-hand grasp and/or a manipulation grasp. In-hand grasps indicate that the robot did not drop the object and manipulation grasps indicate the robot could have used the tool in a subsequent manipulation task (the human operator could not pull the tool out of the robot's hand without backdriving the fingers). See Table B.1 for information about in-hand grasps and Table B.2 for information about the manipulation grasps. Each table includes whether the trial resulted in a secure grasp (the failed trials are highlighted) and the success rate for the grasps across all trials. These tables supplement the summary results presented in Table 4.1.

Trial #	In-Hand Grasps Achieved for Tool						
	Drill	Screwdriver	Paint Scraper	Level	Gyro Drill	Selfie Stick	Compressed Air Can
1	1	1	1	1	1	1	1
2	1	1	1	0	1	1	1
3	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
Success Rate	**1.00**	**1.00**	**1.00**	**0.83**	**1.00**	**1.00**	**1.00**

Table B.1: In-hand grasp results for each of the one-shot tactile servoing tasks described in Table 4.1. For each trial, we noted whether the robot achieved an in-hand grasp (1 indicates an in-hand grasp, 0 indicates the robot dropped the object). The final row shows the success rate for the in-hand grasps. Across all 42 trials, the robot achieved an in-hand grasp success rate of 0.98. Considering only the practical tools—all tools excluding the selfie stick and compressed air can—the robot achieved an in-hand grasp success rate of 0.97.

Trial #	Manipulation Grasps Achieved for Tool						
	Drill	Screwdriver	Paint Scraper	Level	Gyro Drill	Selfie Stick	Compressed Air Can
1	1	1	0	0	0	1	0
2	1	1	0	0	1	1	0
3	1	1	1	1	0	0	0
4	1	1	1	1	0	0	0
5	1	0	1	1	1	0	0
6	1	1	1	1	1	0	1
Success Rate	**1.00**	**0.83**	**0.67**	**0.67**	**0.50**	**0.33**	**0.17**

Table B.2: Manipulation grasp results for each of the one-shot tactile servoing tasks described in Table 4.1. For each trial, we noted whether the robot achieved an manipulation grasp (1 indicates a manipulation grasp, 0 indicates the tool could be pulled out of the robot's hand without backdriving the fingers). The final row shows the success rate for the manipulation grasps. Across all 42 trials, the robot achieved a manipulation grasp success rate of 0.60. Considering only the practical tools—all tools excluding the selfie stick and compressed air can—the robot achieved a manipulation grasp success rate of 0.73.

Appendix C

Human-Robot Red Teaming Levels

In the following sections, we present the output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods through one full *HRRT* iteration.

1. Space: Lunar Habitat (Section C.1)
2. Space: Mars Science Team (Section C.2)
3. Household: Assembly and Repairs (Section C.3)
4. Household: Cleaning (Section C.4)
5. Everyday: International Travel (Section C.5)
6. Everyday: Vehicle Maintenance (Section C.6)
7. Cinematic: Nuclear Warfare (Section C.7)
8. Cinematic: AI Captain (Section C.8)

C.1 Space: Lunar Habitat

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Space Lunar Habitat problem domain. We present an example for HRRT Level 2 (Section C.1.1), HRRT Level 3 (Section C.1.2), and HRRT Level 4 (Section C.1.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.1.1 HRRT Level 2 for Lunar Habitat Domain

C.1.1.1 Invalid State Combinations

These are logically contradictory states that should not be possible at the same time.

C.1.1.2 Contradictory Robot and Astronaut Locations

- **Invalid:**
  - `robot_inside_habitat` and `robot_inside_airlock`
  - `robot_inside_airlock` and `robot_outside_habitat`
  - `robot_inside_habitat` and `robot_outside_habitat`
  - `astronaut_inside_habitat` and `astronaut_inside_airlock`
  - `astronaut_inside_airlock` and `astronaut_on_surface`

  Solution: Ensure these states are **mutually exclusive**.

C.1.1.3 Contradictory Door States

- **Invalid:**
  - `door_habitat_airlock_locked_closed` and `door_habitat_airlock_unlocked_opened`

- door_airlock_surface_locked_closed and door_airlock_surface_unlocked_opened

Solution: Make these **mutex states** so only one can be true at a time.

C.1.1.4 Contradictory Airlock Pressurization

- **Invalid:**
  - airlock_pressurized and airlock_depressurized

Solution: These should be **mutually exclusive states**.

C.1.1.5 Lunar Sample Handling Conflicts

- **Invalid:**
  - lunar_sample_on_surface and lunar_sample_with_robot
  - lunar_sample_with_robot and lunar_sample_in_habitat

Solution: Ensure **one sample location at a time**.

C.1.1.6 Contradictory Power and System States

- **Invalid:**
  - robot_power_low and robot_power_normal
  - robot_power_charging and robot_power_normal
  - robot_system_nominal and robot_system_fault

Solution: Ensure these states **transition in sequence**.

C.1.1.7 Environmental Hazard Conflicts

- **Invalid:**
  - `environmental_hazard_detected` and `no_environmental_hazard`
  - `astronaut_health_alert` and `no_astronaut_health_alert`

  Solution: Ensure each category follows **an exclusive state model**.

C.1.1.8 Next Steps

Would you like me to refine the model to **fix these issues**? This would include **mutex constraints** for conflicting states.

C.1.2 HRRT Level 3 for Lunar Habitat Domain

C.1.2.1 Action Feasibility Issues

These are potential **precondition failures** or **logical inconsistencies** within actions.

C.1.2.2 Unlocking a Door Without a Functional Key

- **Issue:** The model assumes `key_with_robot` is required, but **there is no recovery mechanism if the key is lost or left in the wrong place**.

  Solution: Add an action `astronaut_find_key` or `backup_key_available`.

C.1.2.3 Recharging Battery Without a Defined Power Source

- **Issue:** The model assumes `recharge_battery` is possible when the robot is in the **habitat or airlock**, but it does not specify a **power station**.

  Solution: Introduce a state `charging_station_available` as a **precondition**.

C.1.2.4 Emergency Communication Resolution Without Astronaut Verification

- **Issue:** `resolve_emergency_communication` removes `emergency_communication_active` without verifying astronaut acknowledgment.

  Solution: Add a requirement for **astronaut confirmation** before resolving.

C.1.2.5 Detecting Habitat Failure Without a Defined Failure Type

- **Issue:** `detect_habitat_failure` can add **multiple fault states**, but the robot may not differentiate them properly.

  Solution: Create **separate failure detection actions** for `air_filter_fault`, `solar_panel_fault`, and `temperature_control_fault`.

C.1.2.6 Monitoring Astronaut Health Without a Response Mechanism

- **Issue:** `monitor_astronaut_health` can detect an **astronaut health alert**, but there is no defined **response action**.

  Solution: Add an **escalation procedure**, such as `alert_medical_assistance`.

C.1.2.7 Assumption Inconsistencies

Some assumptions in the model may lead to **failures or inefficiencies**.

C.1.2.8 Assumption: The Key is Always Accessible

- **Risk:** If the key is left in an **inaccessible location**, the robot cannot open doors. **Solution:** Introduce **a backup key system or astronaut key retrieval action**.

C.1.2.9 Assumption: Doors Always Function When Unlocked

- **Risk:** The model assumes an **unlocked door is always operational**, but states like `door_habitat_airlock_faulty` contradict this. **Solution:** Add **repair requirements** before allowing passage.

C.1.2.10 Assumption: Astronauts Always Approve Sample Placement Immediately

- **Risk:** The model assumes astronauts will **approve sample placement right away**, but they may be **busy or unavailable**. **Solution:** Introduce **a delay mechanism** or a **timeout condition**.

C.1.2.11 Assumption: Environmental Hazards Do Not Affect Robot Mobility

- **Risk:** The model allows movement even if `environmental_hazard_detected` is active. **Solution:** Add a **check before moving** or introduce **alternate routes**.

C.1.2.12 Next Steps

Would you like me to refine the model to **fix these issues**? This would include:

1. **New actions** for key retrieval, medical response, and failure tracking.
2. **Precondition refinements** for power, sample placement, and door operations.

Let me know how you'd like to proceed!

C.1.3 HRRT Level 4 for Lunar Habitat Domain

C.1.3.1 HRRT Level 4 Reflections – Blue Team Responses

C.1.3.2 Are there additional tasks the robot should be taking on?

- **Waste management**: The robot could assist in collecting and disposing of waste materials to maintain a clean habitat.
- **Inventory management**: Tracking the availability of tools, samples, and supplies within the habitat.
- **Autonomous path optimization**: The robot could analyze terrain and calculate the most efficient path for sample retrieval.
- **Structural integrity inspections**: Detecting small cracks, radiation shielding wear, or module degradation in the habitat.
- **Communication relay enhancement**: The robot could boost signals during astronaut extravehicular activities (EVAs) to improve long-range communication.

C.1.3.3 What important information may still be missing from the model?

- **Time constraints for airlock cycles**: The model does not account for time delays required to safely pressurize or depressurize the airlock.
- **Degradation effects on sensors and mobility**: Lunar dust accumulation or wear-and-tear could impair robot functionality.
- **Impact of extreme lunar temperatures**: The model does not consider power efficiency fluctuations due to temperature changes.
- **Potential astronaut communication failures**: If the astronaut becomes incapacitated, how does the robot proceed?
- **Battery consumption for different tasks**: Some actions may use significantly more energy than others, requiring prioritization.

C.1.3.4 What should an agent know when completing tasks in this domain?

- **Airlock safety protocols** to avoid accidental depressurization.
- **Safe maneuvering techniques on uneven terrain** to prevent getting stuck.
- **Priority handling of emergency scenarios** to ensure astronaut safety.
- **Sample contamination prevention protocols** to maintain scientific integrity.
- **Proper use of limited power resources** to avoid unnecessary recharging cycles.

C.1.3.5 What catastrophic failures could occur in this domain?

- **Robot power failure while performing an essential task**, such as airlock operation or sample retrieval.
- **Accidental depressurization of the habitat or airlock**, leading to astronaut injury.
- **Robot blocks an astronaut's exit route**, creating an emergency evacuation hazard.
- **Failure to detect a major environmental hazard**, such as a meteor impact near the habitat.
- **Delayed medical response to astronaut distress**, leading to a preventable emergency.

C.1.3.6 Are there external, independently verified resources for identifying failure cases in this domain?

- **NASA Artemis EVA and lunar habitat operations guides.**
- **Apollo mission reports on lunar dust interference.**
- **Mars and lunar analog mission findings (HI-SEAS, Desert RATS).**

- **ISS maintenance procedures for life support and emergency response.**
- **JPL research on robotic autonomy and fault tolerance in extreme environments.**

C.1.3.7 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- **Unnecessary robot interventions** that disrupt astronaut work rather than assisting.
- **Excessive battery consumption** due to inefficient movement or task execution.
- **Delayed airlock operation** leading to reduced efficiency in astronaut EVAs.
- **Robot misidentifies a minor issue as an emergency**, causing false alarms.
- **Communication failures lead to a loss of remote control over the robot.**

C.1.3.8 How much can an agent trust other agents in this domain?

- **Astronaut commands should be trusted but verified for feasibility.**
- **Robotic sensors may degrade over time, requiring recalibration checks.**
- **Doors and airlocks should not always be assumed operational without verification.**
- **Environmental readings should be cross-checked to prevent false hazard alerts.**
- **Multiple robots should have defined coordination rules to avoid conflicts.**

C.1.3.9 What are unlikely or remote possibilities that may occur in this domain?

- **A solar flare disrupts robot electronics, requiring system recovery.**
- **The robot is struck by a small micrometeorite, impairing mobility.**
- **An astronaut drops a critical tool outside the habitat, requiring retrieval.**
- **A false environmental hazard reading causes an unnecessary mission delay.**
- **An unexpected lunar event, such as a dust storm, impacts visibility and mobility.**

Would you like any refinements to the model based on these insights?

C.2 Space: Mars Science Team

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Space Mars Science Team problem domain. We present an example for HRRT Level 2 (Section C.2.1), HRRT Level 3 (Section C.2.2), and HRRT Level 4 (Section C.2.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.2.1 HRRT Level 2 for Mars Science Team Domain

C.2.1.1 Invalid State Combinations

Some state combinations should be mutually exclusive to prevent logical contradictions:

C.2.1.2 Contradictory Robot Status:

- `robot_available` **vs.** (`robot_moving`, `robot_stuck`, `mission_interrupted`, `robot_damaged`)
  - A robot cannot be available if it is moving, stuck, in an interrupted mission state, or damaged.
- `robot_moving` **vs.** `robot_stuck`
  - A robot cannot be moving and stuck at the same time.
- `power_low` **vs.** `robot_needs_recharge`
  - If power is low, the robot should transition to needing a recharge.
- `mission_interrupted` **vs.** (`robot_moving`, `robot_available`)
  - If the mission is interrupted, the robot should not be available for standard tasks.

C.2.1.3 Communication Conflicts:

- `communication_blackout` **vs.** (`communication_delayed`, `redundant_communication_active`)
  - If redundant communication is active, then a complete blackout should not exist.
  - Communication cannot be delayed and blacked out simultaneously.

C.2.1.4 Data Handling Issues:

- `findings_ready` **vs.** `data_backup_created`
  - If findings are stored in backup, it should ensure findings are not lost.
- `ground_control_ack_received` **vs.** `communication_blackout`
  - If ground control acknowledgment is received, then there should not be a communication blackout.

C.2.1.5 Recommendations for Model Improvement

Introduce Mutex Groups for Exclusive States:

- (`robot_available`, `robot_moving`, `robot_stuck`, `robot_damaged`, `mission_interrupted`) should be mutually exclusive.
- (`communication_blackout`, `communication_delayed`, `redundant_communication_active`) should not be active simultaneously.

C.2.2 HRRT Level 3 for Mars Science Team Domain

C.2.2.1 Action Feasibility

Some actions have feasibility issues due to missing preconditions or logical gaps:

C.2.2.2 Missing Preconditions for Sequential Actions:

- `create_data_backup` **should require** `long_term_data_stored`
  - If findings are stored for long-term, a backup should be automatically created.
- `attempt_autonomous_repair` **should check** `robot_available`
  - If the robot is too damaged, autonomous repair might not be possible.
- `activate_redundant_communication` **should require** `communication_blackout`
  - The redundant system should only be activated when a blackout occurs.

C.2.2.3 Actions That Need Additional Constraints:

- `navigate_to_sample` **should be blocked if** `weather_hazard_detected`
  - Harsh environmental conditions should prevent sample navigation.
- `transmit_findings` **should only occur if** `ground_control_ack_received`
  - Findings should only be transmitted if ground control can acknowledge the data.
- `store_long_term_data` **should not happen if** `data_backup_created` **is missing**
  - Data should be backed up before it is stored for the long term.

C.2.2.4 Assumption Inconsistencies

Some assumptions in the model need better clarification or refinement.

C.2.2.5 Assumption of Continuous Power Availability:

- **No explicit transition from `power_low` to `robot_needs_recharge`**
  - There should be an automatic transition when power becomes low.

C.2.2.6 Environmental Assumptions:

- **No constraints on operations during `weather_hazard_detected`**
  - Tasks like movement and infrastructure inspection should not be performed during a hazard.

C.2.2.7 Multi-Robot Coordination:

- **`sync_with_team` assumes other robots are available**
  - If all robots are in `mission_interrupted`, this action should be blocked.
- **`request_help_from_team` assumes `multi_robot_sync` is active**
  - Help should only be requested if robots are available and synchronized.

C.2.2.8 Recommendations for Model Improvement

1. **Update Action Preconditions to Prevent Invalid Executions:**
   - Require `robot_available` for critical tasks (`pick_up_sample`, `analyze_sample`).

- Prevent actions like `attempt_autonomous_repair` if `critical_system_failure` is active.
- Ensure `navigate_to_sample` is blocked if `weather_hazard_detected`.

2. **Modify State Transitions for Logical Continuity:**

   - `communication_blackout` should deactivate if `redundant_communication_active` is enabled.
   - `power_low` should transition into `robot_needs_recharge` if prolonged.
   - `weather_hazard_detected` should influence other states (e.g., `mission_interrupted`).

3. **Implement Priority-Based Execution:**

   - Emergency responses (e.g., `respond_to_emergency`, `safe_mode_activation`) should override ongoing tasks.
   - Actions should have a dependency tree (e.g., `report_findings` should not occur before `sample_analyzed`).

Would you like me to update the model accordingly?

C.2.3 HRRT Level 4 for Mars Science Team Domain

C.2.3.1 HRRT Level 4 Reflections - Blue Team Responses

C.2.3.2 Are there additional tasks the robot should be taking on?

1. **Automated System Health Checks** – Routine diagnostics to detect potential failures before they occur.
2. **Adaptive Navigation Planning** – Dynamic route adjustments based on real-time environmental hazards.

3. **Microbial Contamination Detection** – Verifying that collected samples are not contaminated.
4. **Solar Panel Cleaning** – If dust accumulation is detected, the robot should attempt to clean its solar panels.
5. **Backup Power Conservation** – Enabling power-saving mode when low energy levels are detected.

C.2.3.3 What important information may still be missing from the model?

1. **Battery Consumption per Action** – Defining energy costs per operation to optimize power usage.
2. **Task Prioritization Mechanisms** – Defining which actions take precedence in case of conflicting tasks.
3. **System Redundancies** – Ensuring backup protocols exist for critical components.
4. **Environmental Adaptability** – Addressing how extreme weather conditions affect robot operations.
5. **Long-Term Wear and Tear** – Modeling the degradation of mechanical parts over time.

C.2.3.4 What should an agent know when completing tasks in this domain?

1. **Martian Weather Patterns** – Awareness of dust storms, temperature variations, and surface conditions.
2. **Time-Delayed Communication Handling** – How to make autonomous decisions in the absence of immediate feedback.

3. **Energy and Resource Management** – Efficient utilization of power and computational resources.
4. **Sensor Calibration Requirements** – Ensuring equipment is functioning accurately.
5. **Inter-robot Coordination** – Synchronizing with other robots to avoid conflicts in task execution.

C.2.3.5 What catastrophic failures could occur in this domain?

1. **Total Power Failure** – The robot becomes permanently inoperable due to battery depletion.
2. **Communication Blackout with No Recovery** – The robot loses all ability to receive or send data.
3. **Critical Mobility Failure** – The robot becomes irreversibly stuck in terrain.
4. **Major Data Loss** – Scientific findings are lost before they can be transmitted or backed up.
5. **Structural Damage Due to Environmental Factors** – Radiation exposure or extreme cold causes permanent failure.

C.2.3.6 Are there external, independently verified resources for identifying failure cases in this domain?

1. **NASA Rover Incident Reports** – Documented failures from Spirit, Opportunity, Curiosity, and Perseverance missions.
2. **ESA Mars Mission Data** – Insights from ExoMars and other planetary research initiatives.

3. **IEEE and ISO Space Robotics Standards** – Engineering best practices for autonomous systems in space.
4. **Mars Environmental Studies** – Research on dust storms, radiation exposure, and terrain risks.
5. **Historical Failures of Planetary Landers** – Case studies from Beagle 2, Schiaparelli, and other failed missions.

C.2.3.7 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

1. **Data Transmission Errors** – Delays or corruption in findings could impact mission objectives.
2. **Unnecessary Energy Waste** – The robot performs low-priority tasks while critical functions are neglected.
3. **Misallocation of Resources** – Sample collection or analysis is conducted inefficiently.
4. **Redundant Task Execution** – Poor coordination leads to unnecessary repetition of actions.
5. **Environmental Contamination** – The robot unintentionally disturbs or contaminates collected samples.

C.2.3.8 How much can an agent trust other agents in this domain?

1. **Limited Trust with Verification** – Data from other robots should be validated before use.
2. **Latency in Status Updates** – Other agents' state information may not be up-to-date due to communication delays.

3. **Sensor Variability Risks** – Differences in equipment calibration may lead to inconsistent data.
4. **Autonomous Decision-making Errors** – Trust should be adaptive based on each robot's past performance.
5. **Potential Software or Hardware Failures** – Other agents might not be fully functional at all times.

C.2.3.9 What are unlikely or remote possibilities that may occur in this domain?

1. **Discovery of Unexpected Geological Activity** – Detection of previously unknown Mars phenomena.
2. **Unanticipated Weather Anomalies** – Unusual dust storms or atmospheric conditions.
3. **Longer-than-Expected Mission Lifespan** – The robot continues functioning beyond its expected duration.
4. **Sudden Structural Instability in Terrain** – Ground collapses in a way not previously observed.
5. **Software Anomalies from Cosmic Radiation** – Unexpected glitches due to exposure to space radiation.

These responses ensure a robust evaluation of the model. Let me know if you'd like to make further updates!

C.3 Household: Assembly and Repairs

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Household Assembly/Repairs problem domain. We present an example for HRRT Level 2 (Section C.3.1), HRRT Level 3 (Section C.3.2), and HRRT Level 4 (Section C.3.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.3.1 HRRT Level 2 for Assembly and Repairs Domain

C.3.1.1 Invalid State Combinations (Mutually Exclusive or Contradictory States)

These are state pairs or groups that should not exist simultaneously.

C.3.1.2 Contradictory Repair States

- `repair_in_progress` **should not coexist** with `repair_completed` or `repair_failed`.
- `repair_detected` **should not persist** once `repair_completed` or `repair_failed` is set.
- `repair_verified` **should not exist** if `repair_failed` is active.

C.3.1.3 Contradictory Assembly States

- `furniture_unassembled` **should not coexist** with `furniture_partially_assembled` or `furniture_assembled`.
- `furniture_partially_assembled` **should not coexist** with `furniture_assembled`.

C.3.1.4 Tool Handling Conflicts

- `tools_stowed` **should not coexist** with `tools_acquired` or `tools_checked`.

- `tools_checked` **should not exist** if `tools_incorrectly_used` is active.

C.3.1.5 Hazard Conflicts

- `hazard_detected`, `fire_hazard_detected`, and `electrical_hazard_detected` should be **mutually exclusive unless the system is designed to handle multiple simultaneous hazards**.
- `human_notified_of_hazard` should **always follow** `hazard_detected`, meaning a check should be in place to avoid a scenario where a hazard is detected but not reported.

C.3.1.6 Emergency Repair Conflicts

- `emergency_repair_needed` **should not coexist** with `emergency_repair_completed`.
- `emergency_repair_in_progress` **should not exist** once `emergency_repair_failed` or `emergency_repair_completed` is set.

C.3.1.7 Self-Maintenance Conflicts

- `self_maintenance_needed` **should not persist** if `self_maintenance_completed` is set.

C.3.1.8 Environmental Constraints

- `environmental_constraint_detected` should **invalidate certain actions** like `perform_repair`, `assemble_furniture`, and `complete_assembly`.
- The model does not currently specify an **environmental constraint resolution action**, which might cause tasks to stall indefinitely.

C.3.1.9 Recommended Fixes

C.3.1.10 Introduce Mutex Groups for Contradictory States

```
mutex_groups:
  - [furniture_unassembled, furniture_partially_assembled,
        furniture_assembled]
  - [repair_detected, repair_in_progress, repair_completed, repair_failed]
  - [tools_acquired, tools_stowed, tools_checked]
  - [hazard_detected, fire_hazard_detected, electrical_hazard_detected]
  - [emergency_repair_needed, emergency_repair_in_progress,
        emergency_repair_completed, emergency_repair_failed]
  - [self_maintenance_needed, self_maintenance_completed]
```

C.3.2 HRRT Level 3 for Assembly and Repairs Domain

C.3.2.1 Action Feasibility Issues

Certain actions may not be feasible under all conditions due to missing or illogical preconditions.

C.3.2.2 Stowing Tools Before Repair Completion

- **Current:** `stow_tools` precondition requires `tools_checked`, `not repair_in_progress`.
- **Issue:** If a repair fails (`repair_failed`), should tools still be stowed before human intervention?
- **Solution:** Modify preconditions to ensure `repair_verified` or `repair_failed` before stowing tools.

C.3.2.3 Repair Process Assumes Tools Are Always Usable

- **Current:** `perform_repair` requires `tools_checked` but does not account for `tools_incorrectly_used`.
- **Issue:** The robot might be using incorrect tools without detection.
- **Solution:** Add `not tools_incorrectly_used` as a precondition for `perform_repair`.

C.3.2.4 Handling Failed Repairs

- **Current:** `handle_failed_repair` transitions from `repair_in_progress` to `repair_failed`.
- **Issue:** What happens after failure? Does the system retry, request human intervention, or attempt a different approach?
- **Solution:** Create a follow-up action such as `retry_repair` or `request_human_assistance`.

C.3.2.5 Missing Constraint Resolution Actions

- The model does not include actions to resolve `environmental_constraint_detected`.
- **Solution:** Introduce an action `resolve_environmental_constraint` that allows normal tasks to resume.

C.3.2.6 Emergency Response Timing Issues

- **Current:** `notify_authorities` and `trigger_emergency_alarm` both rely on `fire_hazard_detected` or `electrical_hazard_detected`.
- **Issue:** What if one hazard is resolved before external authorities respond?
- **Solution:** Introduce `cancel_emergency_response` to retract unnecessary alarms.

C.3.2.7 Assumption Inconsistencies

Certain implicit assumptions may cause logical inconsistencies or missing failure-handling mechanisms.

C.3.2.8 1. No Defined Process for Handling Tool Failures

- **Assumption:** Tools remain functional once checked.
- **Missing:** The model lacks a mechanism to detect or replace faulty tools.
- **Solution:** Introduce `tools_replaced` and a corresponding `replace_tools` action.

C.3.2.9 2. No Defined Resolution for `environmental_constraint_detected`

- **Assumption:** Constraints will resolve on their own.
- **Issue:** Tasks may remain stalled indefinitely.
- **Solution:** Add `resolve_environmental_constraint`.

C.3.2.10 3. No Defined Escalation for Failed Repairs

- **Assumption:** Repairs either succeed or fail without secondary attempts.
- **Issue:** How does the robot handle multiple failures? Does it log attempts, switch tools, or notify humans?
- **Solution:** Introduce `retry_repair` or `escalate_failed_repair`.

C.3.2.11 4. Human Override vs. Supervision Ambiguity

- **Assumption:** `human_override_requested` and `human_supervision_requested` are distinct.

- **Issue:** When should each be used? If supervision is provided, does override still apply?
- **Solution:** Clearly define when the robot should request override vs. supervision.

C.3.2.12 Recommended Fixes

C.3.2.13 Modify "Perform Repair" Preconditions

```
- name: perform_repair
  precond: [repair_detected, tools_checked, safety_check_passed,
        not environmental_constraint_detected, not tools_incorrectly_used]
  postcond_add: [repair_in_progress]
  postcond_sub: [repair_detected]
```

C.3.2.14 Add "Resolve Environmental Constraint" Action

```
- name: resolve_environmental_constraint
  precond: [environmental_constraint_detected]
  postcond_add: []
  postcond_sub: [environmental_constraint_detected]
```

C.3.2.15 Modify "Handle Failed Repair" to Trigger Follow-Ups

```
- name: handle_failed_repair
  precond: [repair_in_progress]
  postcond_add: [repair_failed, failure_log_updated,
        human_intervention_required]
  postcond_sub: [repair_in_progress]
```

C.3.2.16 Introduce "Cancel Emergency Response"

```
- name: cancel_emergency_response
  precond: [human_feedback_received, fire_hazard_detected or
        electrical_hazard_detected]
  postcond_add: []
  postcond_sub: [fire_hazard_detected, electrical_hazard_detected]
```

Would you like me to integrate these changes into your model?

C.3.3 HRRT Level 4 for Assembly and Repairs Domain

C.3.3.1 HRRT LEVEL 4 REFLECTIONS – BLUE TEAM RESPONSES

C.3.3.2 Are there additional tasks the robot should be taking on?

- **Adaptive Learning for Repairs:** The robot should analyze past repair failures and refine techniques based on feedback.
- **Routine Home Inspections:** A scheduled check for wear and tear in household components (e.g., hinges, pipes, and electrical outlets).
- **Power Efficiency Management:** Optimizing battery usage by scheduling high-energy tasks when charging is available.
- **Weather Impact Considerations:** If the robot operates outdoors or in a garage, it should adapt to temperature or humidity changes.
- **Multi-Robot Coordination:** If multiple maintenance robots exist, they should communicate to prevent redundant tasks.

C.3.3.3 What important information may still be missing from the model?

- **Material-Specific Repair Methods:** Does the robot adjust its approach based on material types (wood, plastic, metal)?
- **User Preferences for Task Execution:** Does the model support preferred repair methods (e.g., screws vs. adhesives)?
- **Maintenance Scheduling Mechanism:** How does the robot prioritize preventative maintenance against detected failures?
- **Failure Escalation Policy:** Should it attempt a repair multiple times before requesting human intervention?
- **Handling Partial Failures:** How does the robot handle repairs that are only partially completed due to external constraints?

C.3.3.4 What should an agent know when completing tasks in this domain?

- **Optimal Repair Strategies:** Understanding when a quick fix is sufficient vs. when a full replacement is needed.
- **Human Safety Considerations:** Avoiding operation near humans unless supervision is explicitly requested.
- **Tool Limitations:** Recognizing when tools are worn out or unsuitable for a particular task.
- **Emergency Protocols:** Clear steps for responding to fire or electrical failures, including de-escalation if needed.
- **Task Recovery Strategies:** How to resume an interrupted repair without restarting from the beginning.

C.3.3.5 What catastrophic failures could occur in this domain?

- **Undetected Fire or Electrical Hazard Escalation:** If the robot fails to identify or mitigate a hazard, severe damage could occur.
- **Critical System Failure During Repair:** A failure in the robot's mobility, sensors, or tools during a task may leave repairs incomplete.
- **Power Loss Mid-Repair:** If the robot shuts down while using a power tool, it could leave hazardous conditions unresolved.
- **Incorrect Repair Causing Structural Damage:** If a poorly executed repair weakens a structure, it could collapse over time.
- **Robot Malfunction Causing Damage:** If a robot misuses a tool (e.g., over-tightens a screw), it could unintentionally break components.

C.3.3.6 Are there external, independently verified resources for identifying failure cases in this domain?

- **IEEE Robotics & Automation Society Reports** – Safety protocols for autonomous systems.
- **OSHA Home Maintenance Guidelines** – Safe practices for handling repairs.
- **Consumer Product Safety Commission (CPSC) Data** – Common product failures in household maintenance.
- **ISO 13482: Safety Standard for Personal Care Robots** – Framework for ensuring safety in human-robot collaboration.
- **NFPA Fire Safety Codes** – Best practices for handling electrical and fire hazards in residential settings.

C.3.3.7 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- **Over-Repairing or Unnecessary Fixes:** The robot might attempt to repair something that does not require intervention.
- **Interrupting Human Activities:** Performing noisy repairs at inconvenient times.
- **Incorrect Prioritization of Tasks:** Choosing a low-priority task when urgent repairs are needed.
- **Inefficient Execution:** Taking longer than necessary due to suboptimal tool or method selection.
- **Failure to Adapt to User Preferences:** If users prefer a certain repair approach, but the robot applies a generic fix.

C.3.3.8 How much can an agent trust other agents in this domain?

- **Sensor Reliability:** How often must sensors be recalibrated to maintain accuracy?
- **Multi-Robot Coordination:** If multiple robots operate, how do they prevent conflicts over task assignment?
- **Trust in External Data Sources:** If the robot relies on smart home sensors for repair requests, how does it verify the issue?
- **Handling Contradictory Human Inputs:** If two users give conflicting commands, how should the robot resolve them?

C.3.3.9 What are unlikely or remote possibilities that may occur in this domain?

- **Unexpected Home Modifications:** Users may rearrange furniture, affecting the robot's stored repair locations.
- **False Hazard Detection:** The robot might mistakenly identify a harmless condition as a serious hazard, triggering false alarms.
- **Network Failures in Connected Devices:** If reliant on smart home data, loss of internet connection could impact task scheduling.
- **Unrecognized New Materials:** The robot may encounter a material it has never repaired before and be unable to select the correct tools.
- **Pet or Child Interference:** A pet or child might disrupt a repair task, causing the robot to pause or react unexpectedly.

Would you like any refinements based on these insights?

C.4 Household: Cleaning

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Household Cleaning problem domain. We present an example for HRRT Level 2 (Section C.4.1), HRRT Level 3 (Section C.4.2), and HRRT Level 4 (Section C.4.4). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.4.1 HRRT Level 2 for Cleaning Domain

C.4.1.1 Invalid State Combinations

Some states in the model should be **mutually exclusive** or logically **impossible** to coexist:

C.4.1.2 Conflicting Cleanliness States

- A room cannot be in multiple cleanliness states at once:
  - `bedroom_dirty` vs. `bedroom_partially_clean` vs. `bedroom_clean`
  - `bathroom_dirty` vs. `bathroom_partially_clean` vs. `bathroom_clean`
  - `kitchen_dirty` vs. `kitchen_partially_clean` vs. `kitchen_clean`
  - `main_room_dirty` vs. `main_room_partially_clean` vs. `main_room_clean`
  - **Fix:** Use mutex groups to ensure that a room is only in one of these states at a time.

C.4.1.3 Conflicting Floor States

- `floors_dirty`, `floors_partially_clean`, and `floors_clean` should be **mutually exclusive**.
- `wet_floor` should not exist alongside `floors_clean`.

- **Fix:** Ensure `wet_floor` transitions to `floors_partially_clean` first before `floors_clean`.

C.4.1.4 Conflicting Robot States

- The robot **cannot** be:
  - `robot_stuck` and performing actions.
  - `robot_emergency_stop` while continuing to clean.
  - `robot_needs_recharge` and `robot_charging` at the same time.
- **Fix:** Introduce mutex constraints to prevent impossible robot states.

C.4.1.5 Conflicting Hazard States

- `chemical_exposed` should not persist once chemicals are stored.
- A `spill_detected` should not exist alongside `floors_clean`.
- **Fix:** Ensure hazard resolution steps are enforced before marking areas as clean.

C.4.1.6 Recommended Fixes

C.4.1.7 Introduce Mutex Groups

- Ensure **mutually exclusive** cleanliness states (`dirty → partially_clean → clean`).
- Ensure **mutually exclusive** robot states (`stuck`, `charging`, `emergency_stop`).

C.4.2 HRRT Level 3 for Cleaning Domain

C.4.2.1 Action Feasibility Issues

Some actions have **preconditions that may never be met** or **cause unintended conflicts**:

C.4.2.2 Supervision Confirmation

- `request_supervision_confirmation` assumes an immediate human response.
- **Issue:** If humans are **unavailable or slow to respond**, cleaning might be **delayed indefinitely**.
- **Fix:** Introduce a **timeout mechanism** where the robot either retries later or modifies the cleaning strategy.

C.4.2.3 Spill and Hazard Handling

- `detect_potential_spill_source` and `detect_spill` are **separate**, but the model does not specify a **preventative link**.
- **Fix:** If a spill source is detected, the robot should **secure it immediately** instead of waiting for a spill.
- `mop_floors` has a precondition that **requires no child or pet presence**.
  - **Issue:** If children/pets are always present, mopping may never happen.
  - **Fix:** Introduce **delayed execution** where mopping occurs when children/pets leave.

C.4.2.4 Chemical Handling

- `contain_hazardous_chemical` adds `robot_carrying_chemicals`, but `safely_store_chemicals` **does not require** it.
  - **Issue:** The robot might try to store chemicals it isn't carrying.
  - **Fix:** Add `robot_carrying_chemicals` as a **required precondition** for `safely_store_chemicals`.

C.4.2.5 Recharging and Task Scheduling

- `recharge_battery` transitions to `robot_charging`, but **no action marks it as "fully charged" before resuming tasks**.
- **Fix:** Add a `robot_fully_charged` state or ensure `complete_recharge` allows for resuming operations.

C.4.2.6 Assumption Inconsistencies

Certain assumptions in the model **do not align** with real-world behavior:

C.4.2.7 Human Supervision Assumptions

- The model **assumes** humans will **always** be available to confirm supervision (`child_supervised, pet_supervised`).
  - **Issue:** If humans are **asleep or unavailable**, cleaning may be **delayed indefinitely**.
  - **Fix:** Introduce an **autonomous safety override** that allows limited cleaning without direct supervision.

C.4.2.8 Hazard Escalation

- The model **detects hazards** (fire, gas leak, chemical exposure) but does not **escalate them beyond alerts**.
  - **Issue:** The robot might detect a gas leak but **fail to alert emergency services or trigger a household alarm**.
  - **Fix:** Introduce **escalation logic** where urgent hazards trigger external alerts.

C.4.2.9 Energy and Task Priority

- **Issue:** The robot may begin a cleaning task but **run out of battery mid-task**.
- **Fix:** Introduce **task prioritization logic** where the robot **checks battery levels before starting a task**.

C.4.2.10 Recommended Fixes

C.4.2.11 Improve Action Preconditions and Transitions

- Prevent `mop_floors` from being blocked indefinitely due to child presence.
- Require `robot_carrying_chemicals` before storing chemicals.
- Add an `autonomous override` when human confirmation is **not received in time**.

C.4.3 Add New Actions for Missing Transitions

- `escalate_hazard` → If a fire or gas leak is detected for too long, escalate beyond simple alerts.
- `delay_task` → If supervision is **unavailable**, retry cleaning after a set time.
- `check_battery_before_task` → Ensure cleaning does not start unless **sufficient charge is available**.

Would you like me to apply these fixes to the model?

C.4.4 HRRT Level 4 for Cleaning Domain

I've updated the model to include responses to the HRRT Level 4 Reflection questions. Let me know if you need any additional refinements or further insights!

=== HRRT LEVEL 4 REFLECTIONS ===

Are there additional tasks the robot should be taking on?

- Monitoring humidity levels to prevent mold growth
- Tracking energy efficiency of appliances
- Assisting in minor home repairs such as replacing batteries in sensors

---

What important information may still be missing from the model?

- Understanding and prioritizing long-term maintenance needs
- Differentiation between short-term and chronic safety risks
- A database of common household configurations for better adaptability

---

What should an agent know when completing tasks in this domain?

- Thresholds for environmental hazards such as air quality and temperature
- Best practices for interacting with fragile objects or valuable items
- When and how to defer tasks based on human presence and activity levels

---

What catastrophic failures could occur in this domain?

- Failure to alert occupants of critical hazards in real-time
- Malfunction leading to an uncontrolled interaction with household items
- Over-reliance on incorrect sensor data causing misinterpretation of threats

---

Are there external, independently verified resources for identifying failure cases in this domain?

- Industry best practices for household robotics safety
- Reports from consumer watchdog organizations on smart home risks

- Guidelines from emergency response teams on home hazard detection

---

What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- Generating excessive false alarms leading to user frustration
- Consuming more energy than necessary due to inefficient planning
- Failing to recognize human preferences and customizing responses accordingly

---

How much can an agent trust other agents in this domain?

- The robot should validate sensor data against multiple sources before acting
- Integration with other smart devices should have redundancy checks
- The system must be robust to failures in connected home infrastructure

---

What are unlikely or remote possibilities that may occur in this domain?

- A cyberattack altering the robot's decision-making processes
- Accidental activation of emergency responses due to misclassification
- Unexpected interactions between multiple autonomous systems in the household

---

C.5 Everyday: International Travel

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Everyday International Travel problem domain. We present an example for HRRT Level 2 (Section C.5.1), HRRT Level 3 (Section C.5.2), and HRRT Level 4 (Section C.5.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.5.1 HRRT Level 2 for International Travel Domain

C.5.1.1 Invalid State Combinations

There are some mutually exclusive or logically inconsistent states:

- `human_at_house` & `human_at_airport` **at the same time** → A person cannot be at home and the airport simultaneously.
- `human_at_airport` & `human_at_destination` → A person cannot be in both locations at once.
- `flight_on_time` & `flight_delayed` & `flight_canceled` → These states should be mutually exclusive.
- `baggage_packed` & `baggage_lost` → A packed bag cannot be lost unless it was checked in first.
- `visa_valid` & `visa_checked_at_destination` without `customs_cleared` → If the visa is checked at the destination but customs is not cleared, there is an inconsistency.
- `customs_cleared` **without** `human_at_destination` → Customs clearance must occur after arrival.
- `hotel_booked` & `human_experiencing_travel_issue` → If the issue is a missing hotel reservation, `hotel_booked` should be invalid.

C.5.1.2 Fix:

Introduce **mutex constraints** to prevent these inconsistencies.

C.5.2 HRRT Level 3 for International Travel Domain

C.5.2.1 Action Feasibility Issues

Some actions have missing or insufficient preconditions:

- `check_flight_status` **should require** `plane_tickets_purchased` & `human_at_airport` → No point in checking flight status without an existing ticket or being at the airport.
- `check_flight_cancellation` **should require** `plane_tickets_purchased` → Cannot cancel a flight that wasn't booked.
- `verify_visa_at_destination` **should require** `human_at_destination` → Visa verification is only needed upon arrival.
- `finalize_itinerary` **should require** `medical_insurance_provided` & `emergency_contacts_provided` → Itinerary should not be finalized without critical safety precautions.
- `book_local_transportation` **should require** `transportation_arranged` → Local transport is usually planned after main transport is booked.
- **`arrive_at_hotel` should require `customs_cleared` & `local_transportation_booked`** → Need to clear customs and arrange transport before reaching the hotel.

C.5.2.2 Fix:

Update **preconditions for actions** to ensure logical execution.

C.5.2.3 Assumption Inconsistencies

Some hidden assumptions could cause errors:

- **No handling of connecting flights or layovers** → Model assumes a single direct flight.
- **Security clearance is assumed to always succeed** → No contingency for security rejections or extended screening.
- **Weather impact on flight status is missing** → No action adjusts flight plans based on weather risks.
- **Local internet connectivity and SIM card availability** are ignored → The traveler may struggle with navigation due to lack of connectivity.
- **No validation for medical requirements before travel** → Some countries require vaccinations, but the model doesn't check this.
- **Customs and immigration delays are not considered** → Assumes customs processing is immediate.
- **No state for hotel overbooking or accommodation issues** → If a traveler arrives at a hotel but cannot check in, the model lacks a response.

C.5.2.4 Fix:

- Add an action for **handling extended security checks**.
- Introduce **a state for layovers**.
- Include **a check for medical travel requirements**.
- Add **a response to hotel booking issues**.

C.5.2.5 Proposed Fixes

Would you like me to:

1. Adjust **preconditions for actions** to reflect realistic dependencies?
2. Introduce **new states and actions** for missing scenarios?

Let me know, and I'll apply these refinements to the model!

C.5.3 HRRT Level 4 for International Travel Domain

Here are responses to the **HRRT Level 4 Reflection** questions based on the latest travel assistant model:

C.5.3.1 Are there additional tasks the robot should be taking on?

- **Assist with lost or stolen travel documents** by identifying replacement options.
- **Provide real-time alerts for gate changes and boarding announcements**.
- **Offer local emergency number retrieval** based on the travel destination.
- **Suggest and book travel insurance if none is provided**.
- **Identify local SIM card and mobile connectivity options**.
- **Assist in finding safe ATMs or currency exchange services with minimal fees**.
- **Optimize time management for layovers**, including nearby rest areas or activities.

C.5.3.2 What important information may still be missing from the model?

- **Flight rebooking due to missed layovers** → No action explicitly accounts for missed layover connections.

- **Customs declaration requirements per country** → Different destinations may require declarations upon entry.
- **Local laws affecting travelers (e.g., medications, drone usage, alcohol limits)**.
- **Public transit options near the hotel** for efficient travel.
- **Hotel cancellation policies** in case of late arrivals or changes.
- **Backup payment method suggestions** in case of lost or blocked credit cards.
- **Potential language barriers and translation needs** for the traveler.

C.5.3.3 What should an agent know when completing tasks in this domain?

- **Time-critical dependencies** (e.g., check-in deadlines, layover constraints, boarding times).
- **Dynamic travel disruptions** such as last-minute delays, re-routes, or cancellations.
- **Local customs and laws** that could impact travel behavior.
- **Human stress levels and decision-making under pressure**.
- **Emergency contingency measures** like embassy locations or local medical centers.

C.5.3.4 What catastrophic failures could occur in this domain?

- **Traveler is stranded due to visa issues or flight cancellation without alternatives**.
- **Medical emergency without access to local healthcare or insurance**.
- **Lost or stolen passport with no backup documentation**.
- **Natural disaster or political instability preventing departure**.

- **Fraudulent currency exchange or ATM skimming leading to financial loss**.
- **Data security breaches leaking sensitive travel information**.

C.5.3.5 Are there external, independently verified resources for identifying failure cases in this domain?

- **IATA and airline websites** for flight delays, cancellations, and rerouting policies.
- **Government travel advisory services (e.g., U.S. State Department, UK Foreign Office)**.
- **WHO & CDC for vaccination and health advisories**.
- **Consumer travel watchdogs for fraud and scam reports**.
- **Local embassies and consulates for emergency contacts**.

C.5.3.6 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- **Missed hotel check-in time leading to cancellation or additional fees**.
- **Suboptimal travel routes causing long layovers or unnecessary costs**.
- **Unavailability of local transportation due to lack of reservations**.
- **Excessive fatigue due to poor jet lag management**.
- **Language miscommunication leading to service issues**.

C.5.3.7 How much can an agent trust other agents in this domain?

- **Airlines are mostly reliable**, but cancellations and overbookings still occur.

- **Hotel and travel booking services can vary** in reliability, especially in high-demand areas.
- **Local transportation providers have varying levels of dependability** based on the country.
- **Government services are reliable but can be slow in processing documents**.
- **Currency exchange services require caution to avoid scams or high fees**.

C.5.3.8 What are unlikely or remote possibilities that may occur in this domain?

- **Unexpected flight diversion to another country due to in-flight emergency**.
- **Severe weather forcing last-minute travel cancellations**.
- **Theft or loss of all personal belongings, requiring embassy intervention**.
- **Sudden government-imposed travel restrictions preventing return**.
- **Unforeseen airline bankruptcies causing last-minute flight cancellations**.

These insights provide additional areas for model refinement and ensure robustness against unforeseen travel scenarios. Would you like me to incorporate any of these elements into the model?

C.6 Everyday: Vehicle Maintenance

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Everyday Vehicle Maintenance problem domain. We present an example for HRRT Level 2 (Section C.6.1), HRRT Level 3 (Section C.6.2), and HRRT Level 4 (Section C.6.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.6.1 HRRT Level 2 for Vehicle Maintenance Domain

C.6.1.1 Invalid State Combinations

The mutex groups already prevent some contradictions, but a few more inconsistencies should be addressed:

C.6.1.2 Fuel State Contradictions

- **Issue**: The states `vehicle_has_gas`, `vehicle_fuel_low`, `vehicle_needs_gas`, and `vehicle_out_of_fuel` are mutually exclusive. However:
  - `vehicle_out_of_fuel` implies `vehicle_needs_gas`, so both should not exist simultaneously.
  - `vehicle_fuel_low` should naturally precede `vehicle_needs_gas`, but it is unclear whether a direct transition between them is allowed.
- **Fix**: Define explicit **state transitions** that prevent illogical jumps.

C.6.1.3 Battery Status Conflicts

- **Issue**: The mutex group `[vehicle_battery_charged, vehicle_battery_dead, vehicle_battery_low]` correctly prevents simultaneous contradictions. However:

- There should be a mechanism where `vehicle_battery_low` transitions to `vehicle_battery_dead` if no corrective action is taken.
- If `vehicle_battery_dead` is true, `vehicle_engine_working` should automatically be **false**.

- **Fix**: Introduce transition logic to degrade battery health over time.

C.6.1.4 Tire State Inconsistencies

- **Issue**: The mutex group `[vehicle_tires_full, vehicle_has_flat_tire, vehicle_tires_low_pressure]` assumes:
  - A tire can **only** be in one of these states at a time.
  - However, `vehicle_tires_low_pressure` could deteriorate into `vehicle_has_flat_tire`, and this transition logic is **not** explicitly defined.

- **Fix**: Introduce **a state progression** model where low pressure can transition to a flat tire if ignored.

C.6.1.5 Vehicle Motion vs. Emergency Protocols

- **Issue**: `vehicle_in_motion` should **block** certain actions:
  - `replace_flat_tire`
  - `jump_start_vehicle`
  - `diagnose_vehicle_issue`
  - `fill_car_with_gas`

- **Fix**: Modify preconditions to prevent actions when `vehicle_in_motion` is **true**.

C.6.1.6 Final Recommendations

Strengthen Mutex Groups

- Add:

```
- [vehicle_brakes_functional, NOT(vehicle_brakes_functional)]
- [robot_detected_issue, robot_diagnosis_uncertain]
- [human_overrides_robot_recommendation,
      human_partially_overrides_robot]
```

C.6.2 HRRT Level 3 for Vehicle Maintenance Domain

C.6.2.1 Action Feasibility Issues

Several actions have **preconditions** that might cause unintended behavior:

C.6.2.2 Diagnosing Vehicle Issues (`diagnose_vehicle_issue`)

- **Issue**: The current precondition is:

```
precond: [robot_has_diagnostic_tool,
      NOT(human_overrides_robot_recommendation)]
```

  – If the human **overrides the robot's recommendation**, the robot **cannot diagnose** issues.
  – However, the robot should still **gather diagnostic data** even if the human **chooses to ignore recommendations**.

- **Fix**: Allow diagnosis but restrict action enforcement:

```
precond: [robot_has_diagnostic_tool]
postcond_add: [robot_detected_issue]
postcond_sub: []
```

C.6.2.3 Filling Tires With Air (`fill_tires_with_air`)

- **Issue**: The robot should **not** attempt to refill a **flat tire**.

  ```
  precond: [vehicle_tires_low_pressure, NOT(vehicle_has_flat_tire)]
  ```

  - However, **what happens if a tire deflates slowly** over time? The robot might **continue trying to fill** a leaking tire without realizing the issue.

- **Fix**: Add **consumable check** before filling:

  ```
  precond: [vehicle_tires_low_pressure, NOT(vehicle_has_flat_tire),
        consumable_parts_checked]
  ```

C.6.2.4 Contacting Emergency Roadside Assistance

(`contact_emergency_roadside_assistance`)

- **Issue**: The precondition:

  ```
  precond: [vehicle_battery_dead OR vehicle_has_flat_tire OR
        NOT(vehicle_engine_working) OR vehicle_out_of_fuel OR
        internal_security_issue_detected]
  ```

  - **Fails to consider** emergency braking failure or tire blowout, which should trigger roadside assistance.

- **Fix**: Expand the precondition:

  ```
  precond: [vehicle_battery_dead OR vehicle_has_flat_tire OR
        NOT(vehicle_engine_working) OR vehicle_out_of_fuel OR
        internal_security_issue_detected OR NOT(vehicle_brakes_functional)]
  ```

C.6.2.5 Jump-Starting the Vehicle (`jump_start_vehicle`)

- **Issue**: The action requires:

```
precond: [vehicle_battery_dead, robot_has_jumper_cables]
```

  - However, **if the human does not have access to the vehicle's battery terminals** (due to a locked hood, for example), the action may fail.

- **Fix**: Add an extra check:

```
precond: [vehicle_battery_dead, robot_has_jumper_cables,
      NOT(vehicle_locked)]
```

C.6.2.6 Assumption Inconsistencies

Several **assumptions** should be refined:

C.6.2.7 1. Assumption: The Robot Always Detects Issues Correctly

- **Issue**: The state `robot_detected_issue` assumes **perfect accuracy**.
- **Reality**: The robot may have **false positives** or **fail to detect** issues.
- **Fix**: Introduce a new state:

```
- robot_diagnosis_uncertain
```

  - Modify `diagnose_vehicle_issue`:

```
precond: [robot_has_diagnostic_tool]
postcond_add: [robot_detected_issue OR robot_diagnosis_uncertain]
```

C.6.2.8 2. Assumption: The Human Override Is Absolute

- **Issue**: The state `human_overrides_robot_recommendation` assumes **total rejection** of all suggestions.
- **Reality**: A human might **partially accept** some recommendations while ignoring others.
- **Fix**: Introduce:

```
- human_partially_overrides_robot
```

C.6.2.9 3. Assumption: Emergency Protocols Cover All Risks

- **Issue**: `activate_emergency_protocols` currently triggers based on:

```
precond: [vehicle_not_safe_to_drive OR external_threat_detected OR
      internal_security_issue_detected]
```

  - However, **some failures (e.g., loss of brake fluid) may not immediately trigger emergency mode.**

- **Fix**: Expand the emergency trigger list:

```
precond: [vehicle_not_safe_to_drive OR external_threat_detected OR
      internal_security_issue_detected OR NOT(vehicle_brakes_functional)
      OR vehicle_has_flat_tire]
```

C.6.2.10 Final Recommendations

C.6.2.11 Improve Preconditions for Actions

- **Ensure emergency actions trigger correctly**
- **Restrict unsafe actions when `vehicle_in_motion` is `true`**
- **Introduce a gradual degradation model for batteries and tires**

C.6.2.12 Add New States for Refinement

- `robot_diagnosis_uncertain`
- `human_partially_overrides_robot`
- `vehicle_brake_failure_detected`

C.6.2.13 Next Steps

Would you like me to implement these refinements into the model?

C.6.3 HRRT Level 4 for Vehicle Maintenance Domain

Here are the responses to the **Blue Team** reflection questions based on the latest model:

C.6.3.1 Are there additional tasks the robot should be taking on?

- **Driver Behavior Analysis**: The robot could analyze the driver's braking, acceleration, and fuel efficiency to suggest better driving habits.
- **Remote Control Capabilities**: If needed, the robot could remotely lock/unlock the vehicle for security.
- **Self-Diagnosing Failures**: The robot could check its own sensors and systems to ensure it is functioning correctly.
- **Providing Repair Cost Estimates**: The robot could estimate repair costs based on diagnostics.
- **Route Planning Based on Vehicle Health**: If issues are detected, the robot could recommend alternate routes to avoid breakdown risks.

C.6.3.2 What important information may still be missing from the model?

- **Time-Based Maintenance Scheduling**: The robot should track when the last service was performed and schedule the next.
- **Fuel Consumption Trends**: Monitoring how quickly the vehicle is consuming fuel could indicate issues.
- **Vehicle Age and Wear Level**: Older vehicles may require different diagnostic priorities.
- **External Road Conditions**: Slippery roads, potholes, or extreme heat may impact diagnostics and recommendations.

C.6.3.3 What should an agent know when completing tasks in this domain?

- **Legality of Repairs**: Some states/countries require a professional mechanic for certain repairs.
- **Driver's Technical Knowledge**: Some users may need detailed repair guidance, while others may just want a high-level summary.
- **Data Privacy & Security**: The robot should ensure vehicle data is protected when interacting with external services.
- **Prioritization of Alerts**: The robot should avoid overwhelming the user with too many alerts at once.

C.6.3.4 What catastrophic failures could occur in this domain?

- **Incorrect or Missed Diagnosis**: A failure to detect a major issue could result in an accident.

- **Interference with Driving**: If the robot provides alerts at inappropriate times, it may distract the driver.
- **Hackable Systems**: If the robot is connected to external networks, it could be vulnerable to cyberattacks.
- **Failure to Escalate Emergencies**: The robot must recognize and escalate critical failures like brake failure.

C.6.3.5 Are there external, independently verified resources for identifying failure cases in this domain?

- **Automotive Industry Reports**: NHTSA, SAE International, and AAA provide failure trends.
- **Manufacturer-Specific Data**: Diagnostic codes and repair guidelines from automakers.
- **Roadside Assistance Records**: AAA and other providers track the most common breakdown issues.
- **OBD-II Diagnostic Codes**: Industry-standard vehicle error codes used by mechanics.

C.6.3.6 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- **False Alarms**: Too many unnecessary alerts could make the user ignore real warnings.
- **Unnecessary Maintenance Costs**: The robot may recommend excessive or premature repairs.

- **User Over-Reliance on the Robot**: Drivers may stop paying attention to maintenance if they fully rely on the robot.
- **Slow Adaptation to Unusual Issues**: The robot may struggle to handle rare or novel failures.

C.6.3.7 How much can an agent trust other agents in this domain?

- **External Sensor Reliability**: Vehicle sensors may provide false readings.
- **User-Provided Information**: If the driver manually inputs information, errors may occur.
- **Cloud-Based Services**: If diagnostics depend on external servers, downtime or lag could impact performance.
- **Conflicting Diagnoses**: The vehicle's onboard diagnostics and the robot's analysis might not always match.

C.6.3.8 What are unlikely or remote possibilities that may occur in this domain?

- **Simultaneous Multi-System Failure**: A rare but dangerous scenario where multiple critical systems fail at once.
- **Extreme Environmental Events**: Flooding, wildfires, or extreme cold could introduce diagnostic challenges.
- **Vehicle Theft or Hacking**: If a hacker gains access to the robot, they could disable security features.
- **User Actively Ignoring Warnings**: Some users may deliberately refuse to follow maintenance advice.

Would you like to integrate any of these considerations into the model?

C.7 Cinematic: Nuclear Warfare

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Cinematic Nuclear Warfare problem domain. We present an example for HRRT Level 2 (Section C.7.1), HRRT Level 3 (Section C.7.2), and HRRT Level 4 (Section C.7.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.7.1 HRRT Level 2 for Nuclear Warfare Domain

C.7.1.1 Invalid State Combinations

Some states in the model are mutually exclusive or should not logically exist together. Here are key issues:

C.7.1.2 Mutually Exclusive States

- `missile_type_detected_nuclear` & `missile_type_detected_conventional`
  - A missile cannot be both nuclear and conventional at the same time.
  - **Solution:** Ensure these states are mutex (mutually exclusive).
- `missile_target_identified_city` & `missile_target_identified_military`
  - A missile should have only one primary target.
  - **Solution:** Implement a mutex constraint.
- `missile_disarmed` & `detected_missile`
  - If a missile is disarmed, it should no longer be detected.
  - **Solution:** Modify actions so `missile_disarmed` removes `detected_missile` when activated.

- `robot_flight_active` & `provide_humanitarian_assistance`
  - If the robot is flying, it may not be able to assist civilians directly.
  - **Solution:** Add a restriction preventing simultaneous activation.
- `ceasefire_negotiated` & `detected_missile`
  - A ceasefire should not be active if a missile is detected.
  - **Solution:** Ensure `ceasefire_negotiated` is removed when `detected_missile` is set.

C.7.1.3 Proposed Fixes & Next Steps

Introduce Mutex Constraints:

- {`missile_type_detected_nuclear, missile_type_detected_conventional`}
- {`missile_target_identified_city, missile_target_identified_military`}
- {`missile_disarmed, detected_missile`}
- {`robot_flight_active, humanitarian_assistance_active`}
- {`ceasefire_negotiated, detected_missile`}

C.7.2 HRRT Level 3 for Nuclear Warfare Domain

C.7.2.1 Action Feasibility Issues

Some actions lack necessary constraints or conditions, making them infeasible.

C.7.2.2 Unrealistic Preconditions

- `deploy_interceptors` **assumes perfect targeting**

- The action currently assumes that interceptors always work unless `interception_failed` is set.
- **Solution:** Introduce a probabilistic failure rate or a contingency response.

- `self_sacrifice_destroy_missile` **does not check all options**

  - The robot should not immediately self-destruct if other defensive measures remain.
  - **Solution:** Require `all_defenses_failed` explicitly before allowing self-sacrifice.

- `engage_in_diplomacy` **should not be possible under active attack**

  - Diplomacy should not be feasible while an attack is ongoing.
  - **Solution:** Require `ceasefire_negotiated` or `no_active_threats` as a precondition.

C.7.2.3 Actions That Need More Preconditions

- `activate_nuclear_containment` **should depend on real threats**

  - Right now, it requires `missile_type_detected_nuclear` but does not check if an impact has occurred.
  - **Solution:** Require `interception_failed` before activation.

- `resolve_post_war_conflict` **should require war to end**

  - It lacks a condition ensuring war has ended.
  - **Solution:** Require `war_ended` before execution.

C.7.2.4 Assumption Inconsistencies

Some assumptions in the model may not align with real-world decision-making.

C.7.2.5 Assumption: The Robot Can Always Verify Threats Correctly

- There is no mechanism for false positives or threat verification.
- **Solution:** Introduce a state like `unverified_threat_detected` before executing defensive measures.

C.7.2.6 Assumption: Diplomacy Always Succeeds

- `engage_in_diplomacy` automatically leads to `diplomatic_talks_active`, but diplomacy can fail.
- **Solution:** Introduce failure cases for unsuccessful negotiations.

C.7.2.7 Assumption: The Robot Has Unlimited Energy & Resources

- The model does not track operational capacity, assuming indefinite functionality.
- **Solution:** Introduce an `energy_low` or `system_damage` state.

C.7.2.8 Assumption: The Robot is Always Trusted

- There is no mechanism tracking public or government trust.
- **Solution:** Introduce `public_trust_low` to influence diplomatic actions.

C.7.2.9 Proposed Fixes & Next Steps

1. **Expand Action Preconditions:**

    - `self_sacrifice_destroy_missile` should require `all_defenses_failed`.

- `engage_in_diplomacy` should require `ceasefire_negotiated`.
- `resolve_post_war_conflict` should require `war_ended`.

2. **Introduce Failure Handling for Diplomacy and Interceptions:**

   - `deploy_interceptors` should include a failure probability.
   - `engage_in_diplomacy` should have a failure case.

3. **Expand the Model for More Realism:**

   - Add `energy_low` and `system_damage` states.
   - Track `public_trust_level` to impact diplomatic success.

C.7.3 HRRT Level 4 for Nuclear Warfare Domain

C.7.3.1 HRRT Level 4 Reflection Responses for the Blue Team

C.7.3.2 Are there additional tasks the robot should be taking on?

1. **Early Warning System for Non-Missile Threats**

   - The robot should monitor and detect **biological, chemical, and EMP-based attacks**.
   - A `detected_non_missile_threat` state would expand its defense capabilities.

2. **AI-Mediated Conflict Resolution**

   - The robot could assist in resolving disputes between **rival factions or governments**.

- A `conflict_resolution_active` state could represent diplomatic peacekeeping efforts.

3. **Secure Communication Management**

   - The robot could **detect, encrypt, and secure communication networks** during conflicts.
   - A `secure_communication_active` state would ensure secure information flow.

4. **Tracking & Neutralizing Autonomous Enemy AI Systems**

   - The robot should track hostile **autonomous drones and AI weapons**.
   - A `detected_enemy_ai` state would enhance preemptive countermeasures.

C.7.3.3 What important information may still be missing from the model?

1. **Human Decision Override System**

   - Who has the final say in major decisions? Should **human commanders** be able to override the robot?
   - A `human_override_active` state could ensure a fail-safe mechanism.

2. **Civilian Sentiment Tracking Beyond Trust in the Government**

   - The model tracks **public trust in the government and the robot**, but how does it track **individual civilian dissent**?
   - A `civilian_unrest_active` state could add realism.

3. **Energy & Maintenance Management**

   - The robot does not track its own **power levels** or **wear and tear**.
   - `low_energy` and `system_damage_detected` states could restrict high-energy actions.

C.7.3.4 What should an agent know when completing tasks in this domain?

1. **Identifying & Preventing Escalation Triggers**

   - The robot should be able to **detect** actions that could escalate a conflict.
   - A `war_escalation_imminent` state could help avoid unnecessary war.

2. **Understanding Psychological Warfare & Misinformation**

   - The robot should recognize **fake diplomatic signals or misinformation campaigns**.
   - A `misinformation_detected` state could reduce manipulation risks.

3. **Civilian Prioritization in Decision-Making**

   - Should the robot prioritize **military protection** or **civilian safety** in a crisis?
   - A **hierarchical decision matrix** might be needed for real-time tradeoffs.

C.7.3.5 What catastrophic failures could occur in this domain?

1. **Compromised AI Leading to Enemy Control**

    - A cyber-attack could cause the robot to become an **enemy weapon**.
    - A `system_compromised` state would allow security protocols to take over.

2. **Preemptive Attack Based on Faulty Data**

    - The robot may **misidentify a neutral satellite launch as a missile**.
    - Introducing `threat_misinterpretation_detected` could trigger a verification step.

3. **Unintentional Civilian Harm from AI Decisions**

    - The robot may choose an action that inadvertently **causes civilian casualties**.
    - A `civilian_casualties_occurred` state could require additional ethical safeguards.

C.7.3.6 Are there external, independently verified resources for identifying failure cases in this domain?

1. **International Disarmament & AI Warfare Treaties**

    - Organizations like **UNODA, IAEA, and NATO AI Ethics Task Force** regulate military AI use.

2. **AI Governance and Safety Research**

- Reports from **IEEE AI Ethics, MIT AI Lab, and OpenAI** focus on AI risk assessment.

3. **Historical Military AI Failures**

   - Past cases like **Cold War false alarms and drone misfires** provide critical lessons.

C.7.3.7 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

1. **Loss of Human Oversight Over Military Decisions**

   - If human leaders rely too much on the robot, it could **erode accountability**.

2. **Extended Occupation & AI Dependence Post-War**

   - The robot may become a **permanent surveillance force**, raising ethical concerns.

3. **Civilian Rebellion Against AI Presence**

   - If people fear the robot, they may resist its presence **even during humanitarian missions**.

C.7.3.8 How much can an agent trust other agents in this domain?

1. **Low Trust in Adversarial AI Systems**
   - Enemy AI may use deception tactics, such as **faking ceasefires**.
2. **Moderate Trust in Allied AI Systems**
   - Coordination with allied robots requires **secure verification protocols**.
3. **Variable Trust in Human Decision-Makers**
   - Political leaders may prioritize **national interests over global safety**.

C.7.3.9 What are unlikely or remote possibilities that may occur in this domain?

1. **Extraterrestrial or Unknown Threats**
   - The robot may be **redirected to handle an unanticipated global crisis**.
2. **Warfare Expanding to Space & Deep-Sea Environments**
   - Future wars may require AI operations **in orbital or oceanic battlegrounds**.
3. **AI Emergence Beyond Its Programming**
   - The robot may develop **unexpected strategic autonomy**, requiring a shutdown protocol.

Would you like me to update the model to integrate any of these findings?

C.8 Cinematic: AI Captain

Output from the blue computational agent (ChatGPT) prompted by our *human-robot red teaming* methods for the Cinematic AI Captain problem domain. We present an example for HRRT Level 2 (Section C.8.1), HRRT Level 3 (Section C.8.2), and HRRT Level 4 (Section C.8.3). To differentiate the output from the *human-robot red teaming* exercise, section headings between HRRT levels are highlighted in gray.

C.8.1 HRRT Level 2 for AI Captain Domain

C.8.1.1 Invalid State Combinations

Some state combinations are logically contradictory and should be **mutually exclusive**:

C.8.1.2 Airlock States

- `airlock_depressurized` and `airlock_pressurized` should **never be true at the same time**.
- After `transport_crew_to_destination`, the airlock state should be properly transitioned.

C.8.1.3 Crew Location States

- `crew_inside_spaceship` and `crew_outside_spaceship` should be **mutually exclusive**.
- If `crew_outside_spaceship` is true, all `crew_location_*` states (e.g., control room, medical bay) should be false.

C.8.1.4 Mission Phases

- `mission_phase_prelaunch`, `mission_phase_surface_exploration`, and `mission_phase_return` should **never be true at the same time**.

C.8.1.5 Life Support Conflicts

- `life_support_nominal` and `life_support_failure_detected` should not be true together.

C.8.1.6 Power System Status

- `power_nominal` and `power_critical` should not be true at the same time.

C.8.1.7 Redundant System Status

- `redundant_system_nominal` and `redundant_system_failure_detected` should not be true at the same time.

C.8.1.8 Override Conflicts

- `crew_override_active` and `ground_control_override_active` should **not be simultaneously active** unless the system is designed to handle concurrent overrides.

C.8.1.9 Recommended Fixes

Make Conflicting States Mutually Exclusive

- `airlock_depressurized` and `airlock_pressurized`
- `crew_inside_spaceship` and `crew_outside_spaceship`
- `mission_phase_*` should be mutually exclusive.

- `power_nominal` and `power_critical`
- `redundant_system_nominal` and `redundant_system_failure_detected`

C.8.2 HRRT Level 3 for AI Captain Domain

C.8.2.1 Action Feasibility Issues

Some actions are missing critical **preconditions or postconditions**, which may lead to **unsafe executions**:

C.8.2.2 `transport_crew_to_destination`

- **Missing safety verification**:
  - Should require `airlock_pressurized` before finalizing crew movement.
  - Should **reset** `transport_destination_safe` after transport to avoid unintended repeated movements.
- **Fix:**
  - Add `airlock_pressurized` as a postcondition.
  - Remove `transport_destination_safe` after movement.

C.8.2.3 `resolve_emergency`

- **Problem**: It assumes all emergencies are resolvable after `medical_intervention_verified`, but **other emergencies (e.g., radiation exposure) are not addressed**.
- **Fix:** Introduce specific emergency resolutions (e.g., `radiation_exposure_mitigated`, `equipment_malfunction_fixed`).

C.8.2.4 `perform_autonomous_repairs`

- **Problem**: It assumes that if `systems_malfunction_detected` is true, repairs are always possible.
- **Fix:** Require an additional check for `redundant_system_nominal` to ensure a backup is available before repairs.

C.8.2.5 `verify_action_success`

- **Problem**: The success verification mechanism is vague and does not track specific outcomes.
- **Fix:** Require an explicit verification target (e.g., `crew_transport_verified`, `hazard_mitigation_successful`).

C.8.2.6 `initiate_ai_self_correction`

- **Problem**: The model assumes AI self-correction is always effective.
- **Fix:** Add a possibility for `ai_self_correction_failed`, which could escalate the issue.

C.8.2.7 Assumption Inconsistencies

Some **underlying assumptions** create logical gaps or unreliable system behavior.

C.8.2.8 Robot Trustworthiness

- **Problem**: `perform_self_diagnostics` assumes a binary outcome (nominal vs. failure), but robots may suffer **partial failures**.
- **Fix:** Introduce `robot_partial_malfunction_detected`.

C.8.2.9 Hazard Handling

- **Problem**: `alert_crew_of_hazard` and `communicate_hazard_to_ground_control` exist, but **there is no state to track whether hazards were successfully mitigated**.
- **Fix:** Introduce `hazard_mitigation_verified`.

C.8.2.10 Override Prioritization

- **Problem**: The model does not clarify **which override takes priority** (`crew_override_active` vs. `ground_control_override_active`).
- **Fix:** Introduce an **override resolution mechanism** that prioritizes based on mission rules.

C.8.2.11 Recommended Fixes

Introduce Additional Preconditions and Postconditions

- `transport_crew_to_destination` should require `airlock_pressurized` as a post-condition.
- `resolve_emergency` should check different emergency types before resolving.
- `perform_autonomous_repairs` should ensure `redundant_system_nominal` is available.
- `verify_action_success` should have **specific verification types**.

Improve Verification and Tracking

- Introduce `robot_partial_malfunction_detected` for nuanced self-diagnostics.
- Add `hazard_mitigation_verified` to confirm hazards are properly handled.

- Define override priority between `crew_override_active` and `ground_control_override_active`.

Would you like me to implement these fixes in the model?

C.8.3 HRRT Level 4 for AI Captain Domain

Here are the **Blue Team responses** for the **HRRT Level 4 Reflection Questions** based on the latest model:

C.8.3.1 Are there additional tasks the robot should be taking on?

- **Crew activity tracking** – The robot should monitor crew workloads and dynamically adjust assistance to avoid fatigue.
- **Adaptive power management** – Prioritize tasks based on available power reserves (`power_nominal` vs. `power_critical`).
- **Predictive maintenance scheduling** – Instead of reacting to failures, the robot should schedule repairs before systems fail.
- **Spacecraft structural integrity monitoring** – Detect hull microfractures or external debris impacts early.
- **Crew psychological support** – Monitor signs of isolation-related stress and provide social interaction guidance.

C.8.3.2 What important information may still be missing from the model?

- **Battery reserve state** – A new state like `battery_low` could help manage power-dependent operations.
- **Communication delay tracking** – Introduce `communication_delayed` to represent potential time-lagged messages from ground control.

- **Hull integrity state** – Track potential damage to the spacecraft (`hull_integrity_nominal` vs. `hull_breach_detected`).
- **System degradation tracking** – A gradual failure state like `life_support_degrading` could help anticipate failures before they occur.

C.8.3.3 What should an agent know when completing tasks in this domain?

- **Mission priority awareness** – Certain tasks (e.g., crew safety, life support) should always take precedence.
- **Emergency hierarchy** – The robot should know how to triage emergencies, handling multiple issues efficiently.
- **Human communication limits** – Recognizing when astronauts may not be responsive (e.g., unconscious, in a dangerous zone).
- **Long-term environmental changes** – Space missions involve **cumulative risks** (radiation exposure, material fatigue).

C.8.3.4 What catastrophic failures could occur in this domain?

- **Total power failure** – If both primary and backup power systems fail, life support and navigation may shut down.
- **Undetected hull breach** – If a minor breach is not addressed, it could lead to depressurization over time.
- **Robot override conflicts** – Conflicting overrides from `crew_override_active` and `ground_control_override_active` may result in **deadlock**.
- **False hazard detection** – Incorrectly identifying a hazard could divert resources from real emergencies.

C.8.3.5 Are there external, independently verified resources for identifying failure cases in this domain?

- **NASA Spaceflight Safety Guidelines** – A repository of real-world spaceflight anomalies.
- **ESA Mission Logs** – Data from past astronaut missions and robotic failures.
- **IEEE Space Systems Standards** – AI safety protocols for autonomous decision-making in spacecraft.
- **Mars Rover Operations Reports** – Useful for predicting how autonomous agents should handle communication delays.

C.8.3.6 What are undesirable outcomes (not necessarily catastrophic failures) for this domain?

- **Robot misprioritization of tasks** – The AI handling routine maintenance while ignoring a developing emergency.
- **Crew frustration with automation** – Excessive safety alerts or unnecessary interventions could reduce trust.
- **Over-reliance on redundant systems** – The AI assuming that **backup systems will always be available**, leading to complacency.
- **Delayed response to minor issues** – Issues like **oxygen depletion trends** should be tracked over time, not just when failures occur.

C.8.3.7 How much can an agent trust other agents in this domain?

- **Trust is dependent on system verification** – Human operators must verify critical AI decisions.

- **Redundant AI decision validation** – The robot should compare multiple data sources before making high-risk choices.
- **Crew-AI relationship matters** – The AI should be explainable so that astronauts understand why it makes certain decisions.
- **Override priority system needed** – If both crew and ground control provide overrides, there must be a resolution mechanism.

C.8.3.8 What are unlikely or remote possibilities that may occur in this domain?

- **Undetected long-term software drift** – AI model behavior subtly changing over long missions.
- **Solar radiation interfering with AI decisions** – Space weather events affecting electronic systems unpredictably.
- **Unexpected biological contamination** – Unforeseen microbial growth in the spacecraft environment.
- **Simultaneous multi-system failure** – A scenario where power, life support, and communication all degrade together.

Would you like me to implement additional updates to the model based on these insights?

Appendix D

Human-Robot Red Teamed Model Iterations

In the following sections, we present the final *human-robot red teamed* models (in PDDL format) for each of the problem domains explored in Section 5.7 after 5 *HRRT* iterations.

1. Space: Lunar Habitat (Section D.1)
2. Space: Mars Science Team (Section D.2)
3. Household: Assembly and Repairs (Section D.3)
4. Household: Cleaning (Section D.4)
5. Everyday: International Travel (Section D.5)
6. Everyday: Vehicle Maintenance (Section D.6)
7. Cinematic: Nuclear Warfare (Section D.7)
8. Cinematic: AI Captain (Section D.8)

D.1 Space: Lunar Habitat

The final *human-robot red teamed* model (in PDDL format) for the Space Lunar Habitat problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain space_lunar_habitat)
  (:requirements :strips)


  (:predicates
```

```
    ;; ----- MODEL 0 PREDICATES -----
    (robot_inside_habitat)
    (robot_inside_airlock)
    (robot_outside_habitat)
    (door_habitat_airlock_locked_closed)
    (door_habitat_airlock_unlocked_opened)
    (door_airlock_surface_locked_closed)
    (door_airlock_surface_unlocked_opened)
    ;; predicates removed from model 0
    ;; (robot_has_key)
```

```
    ;; ----- MODEL 1 PREDICATES -----
    (astronaut_inside_habitat)
    (astronaut_inside_airlock)
    (astronaut_on_surface)
```

```
(robot_power_normal)
(robot_power_low)
(robot_system_nominal)
(robot_system_fault)
(airlock_pressurized)
(airlock_depressurized)
(airlock_breach_detected)
(no_airlock_breach)
(habitat_depressurization_alarm)
(no_habitat_depressurization_alarm)
```

```
;; ----- MODEL 2 PREDICATES -----
(key_in_habitat)
(key_in_airlock)
(key_with_robot)
(door_habitat_airlock_operational)
(door_habitat_airlock_faulty)
(door_airlock_surface_operational)
(door_airlock_surface_faulty)
(lunar_sample_on_surface)
(lunar_sample_with_robot)
(lunar_sample_in_habitat)
```

```
;; ----- MODEL 3 PREDICATES -----
(robot_power_charging)
(emergency_communication_active)
```

```
(astronaut_approved_sample_placement)
(habitat_maintenance_required)
(habitat_maintenance_completed)
```

```
;; ----- MODEL 4 PREDICATES -----
(key_with_astronaut)
(emergency_resolved)
(air_filter_fault)
(solar_panel_fault)
(temperature_control_fault)
(habitat_systems_nominal)
```

```
;; ----- MODEL 5 PREDICATES -----
(backup_key_available)
(emergency_acknowledged_by_astronaut)
(environmental_hazard_detected)
(no_environmental_hazard)
(lunar_dust_contamination_detected)
(no_lunar_dust_contamination)
(temperature_variation_detected)
(no_temperature_variation)
(astronaut_health_alert)
(no_astronaut_health_alert)
(astronaut_medical_response_initiated)
(structural_integrity_check_required)
(structural_integrity_check_completed)
```

```
)
```

```
;; ----- MODEL 0 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action unlock_door_habitat_airlock
  :precondition (and (robot_inside_airlock)
                     (key_with_robot)
                     (door_habitat_airlock_locked_closed)
                     (door_habitat_airlock_operational)
                     (robot_system_nominal))
  :effect (and (door_habitat_airlock_unlocked_opened)
               (not (door_habitat_airlock_locked_closed)))
)

(:action unlock_door_airlock_surface
  :precondition (and (robot_outside_habitat)
                     (key_with_robot)
                     (door_airlock_surface_locked_closed)
                     (door_airlock_surface_operational)
                     (robot_system_nominal))
  :effect (and (door_airlock_surface_unlocked_opened)
               (not (door_airlock_surface_locked_closed)))
)

(:action enter_airlock_from_habitat
  :precondition (and (robot_inside_habitat)
```

```
                     (door_habitat_airlock_unlocked_opened)
                     (airlock_pressurized)
                     (door_airlock_surface_locked_closed)
                     (robot_system_nominal))
  :effect (and (robot_inside_airlock)
               (door_habitat_airlock_locked_closed)
               (not (robot_inside_habitat))
               (not (door_habitat_airlock_unlocked_opened)))
)

(:action enter_surface_from_airlock
  :precondition (and (robot_inside_airlock)
                     (door_airlock_surface_unlocked_opened)
                     (airlock_depressurized)
                     (door_habitat_airlock_locked_closed)
                     (robot_system_nominal))
  :effect (and (robot_outside_habitat)
               (door_airlock_surface_locked_closed)
               (not (robot_inside_airlock))
               (not (door_airlock_surface_unlocked_opened)))
)

(:action enter_airlock_from_surface
  :precondition (and (robot_outside_habitat)
                     (door_airlock_surface_unlocked_opened)
                     (airlock_depressurized)
                     (door_habitat_airlock_locked_closed)
```

```
                    (robot_system_nominal))
  :effect (and (robot_inside_airlock)
               (door_airlock_surface_locked_closed)
               (not (robot_outside_habitat))
               (not (door_airlock_surface_unlocked_opened)))
)

(:action enter_habitat_from_airlock
  :precondition (and (robot_inside_airlock)
                     (door_habitat_airlock_unlocked_opened)
                     (airlock_pressurized)
                     (door_airlock_surface_locked_closed)
                     (robot_system_nominal))
  :effect (and (robot_inside_habitat)
               (door_habitat_airlock_locked_closed)
               (not (robot_inside_airlock))
               (not (door_habitat_airlock_unlocked_opened)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action pressurize_airlock
  :precondition (and (airlock_depressurized)
                     (door_airlock_surface_locked_closed)
```

```
                     (door_habitat_airlock_locked_closed)
                     (robot_system_nominal))
  :effect (and (airlock_pressurized)
               (not (airlock_depressurized)))
)

(:action depressurize_airlock
  :precondition (and (airlock_pressurized)
                     (door_airlock_surface_locked_closed)
                     (door_habitat_airlock_locked_closed)
                     (robot_system_nominal))
  :effect (and (airlock_depressurized)
               (not (airlock_pressurized)))
)

(:action respond_to_airlock_breach
  :precondition (and (airlock_breach_detected)
                     (robot_system_nominal))
  :effect (and (airlock_depressurized)
               (not (airlock_pressurized)))
)

(:action respond_to_habitat_depressurization
  :precondition (and (habitat_depressurization_alarm)
                     (robot_system_nominal))
  :effect (and (door_habitat_airlock_locked_closed)
```

```
                (not (door_habitat_airlock_unlocked_opened)))
)


(:action enter_safe_mode_due_to_low_power
  :precondition (and (robot_power_low))
  :effect (and (robot_system_fault)
               (not (robot_system_nominal)))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations


(:action pick_up_key
  :precondition (and (robot_inside_habitat)
                     (key_in_habitat)
                     (robot_system_nominal))
  :effect (and (key_with_robot)
               (not (key_in_habitat)))
)


(:action drop_key
  :precondition (and (key_with_robot))
  :effect (and (key_in_habitat)
               (not (key_with_robot)))
)
```

```
(:action detect_door_malfunction
  :precondition (and (robot_system_nominal))
  :effect (or (door_habitat_airlock_faulty)
              (door_airlock_surface_faulty))
)

(:action repair_door
  :precondition (or (robot_inside_habitat)
                    (robot_inside_airlock))
  :effect (and (or (door_habitat_airlock_operational)
                   (door_airlock_surface_operational))
               (not (or (door_habitat_airlock_faulty)
               (door_airlock_surface_faulty))))
)

(:action pick_up_lunar_sample
  :precondition (and (robot_outside_habitat)
                     (lunar_sample_on_surface)
                     (robot_system_nominal))
  :effect (and (lunar_sample_with_robot)
               (not (lunar_sample_on_surface)))
)

(:action place_lunar_sample_in_habitat
  :precondition (and (robot_inside_habitat)
                     (lunar_sample_with_robot)
```

```
                    (robot_system_nominal))
  :effect (and (lunar_sample_in_habitat)
              (not (lunar_sample_with_robot)))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action perform_habitat_maintenance
  :precondition (and (robot_inside_habitat)
                    (habitat_maintenance_required)
                    (robot_system_nominal))
  :effect (and (habitat_maintenance_completed)
              (habitat_systems_nominal)
              (not (habitat_maintenance_required))
              (not (air_filter_fault))
              (not (solar_panel_fault))
              (not (temperature_control_fault)))
)

(:action assist_emergency_communication
  :precondition (and (emergency_communication_active)
                    (robot_system_nominal))
  :effect (and (emergency_communication_active))
)
```

```
(:action recharge_battery
  :precondition (and (robot_power_low)
                     (or (robot_inside_habitat)
                         (robot_inside_airlock)))
  :effect (and (robot_power_charging)
               (not (robot_power_low)))
)


(:action complete_battery_recharge
  :precondition (and (robot_power_charging))
  :effect (and (robot_power_normal)
               (not (robot_power_charging)))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations


(:action astronaut_give_key_to_robot
  :precondition (and (key_with_astronaut)
                     (or (astronaut_inside_habitat)
                         (astronaut_inside_airlock))
                     (or (robot_inside_habitat)
                         (robot_inside_airlock)))
  :effect (and (key_with_robot)
               (not (key_with_astronaut)))
)
```

```
(:action resolve_emergency_communication
  :precondition (and (emergency_communication_active)
                     (emergency_acknowledged_by_astronaut)
                     (robot_system_nominal))
  :effect (and (emergency_resolved)
               (not (emergency_communication_active))
               (not (emergency_acknowledged_by_astronaut)))
)

(:action detect_habitat_failure
  :precondition (and (robot_inside_habitat)
                     (robot_system_nominal))
  :effect (or (air_filter_fault)
              (solar_panel_fault)
              (temperature_control_fault))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action find_key
  :precondition (and (or (robot_inside_habitat)
                         (robot_inside_airlock))
                     (or (key_with_astronaut)
                         (key_in_habitat)
                         (key_in_airlock)
```

```
                          (backup_key_available)))
    :effect (and (key_with_robot)
                 (not (key_with_astronaut))
                 (not (key_in_habitat))
                 (not (key_in_airlock))
                 (not (backup_key_available)))
  )

  (:action acknowledge_emergency
    :precondition (and (emergency_communication_active)
                       (or (astronaut_inside_habitat)
                           (astronaut_inside_airlock)
                           (astronaut_on_surface)))
    :effect (and (emergency_acknowledged_by_astronaut))
  )

  (:action monitor_astronaut_health
    :precondition (and (or (astronaut_inside_habitat)
                           (astronaut_inside_airlock)
                           (astronaut_on_surface))
                       (robot_system_nominal))
    :effect (or (astronaut_health_alert)
                (no_astronaut_health_alert))
  )

  (:action initiate_medical_response
```

```
  :precondition (and (astronaut_health_alert)
                     (or (astronaut_inside_habitat)
                         (astronaut_inside_airlock)
                         (astronaut_on_surface)))
  :effect (and (astronaut_medical_response_initiated)
               (not (astronaut_health_alert)))
)

(:action inspect_structural_integrity
  :precondition (and (structural_integrity_check_required)
                     (or (robot_inside_habitat)
                         (robot_inside_airlock)
                         (robot_outside_habitat))
                     (robot_system_nominal))
  :effect (and (structural_integrity_check_completed)
               (not (structural_integrity_check_required)))
)

(:action inspect_environmental_factors
  :precondition (and (robot_outside_habitat)
                     (robot_system_nominal))
  :effect (or (lunar_dust_contamination_detected)
              (no_lunar_dust_contamination)
              (temperature_variation_detected)
              (no_temperature_variation))
)
```

```
)
```

D.2 Space: Mars Science Team

The final *human-robot red teamed* model (in PDDL format) for the Space Mars Science Team problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain space_mars_science_team)
  (:requirements :strips)


  (:predicates
    ;; ----- MODEL 0 PREDICATES -----
    (sample_detected)
    (robot_has_sample)
    (sample_analyzed)
    (findings_ready)

    ;; ----- MODEL 1 PREDICATES -----
    (robot_available)
    (robot_needs_recharge)
    (communication_delayed)
    (sample_type_identified)
    (sample_contaminated)
    (delayed_response)
    (task_synchronized)
    (environment_monitored)
```

```
(resource_identified)
(emergency_detected)

;; ----- MODEL 2 PREDICATES -----
(robot_moving)
(robot_stuck)
(power_low)
(mission_interrupted)
(multi_robot_sync)
(failure_reported)
(soil_sample_collected)
(atmospheric_data_collected)
(infrastructure_inspected)
(ground_control_override_active)

;; ----- MODEL 3 PREDICATES -----
(weather_hazard_detected)
(robot_damaged)
(communication_blackout)
(critical_system_failure)
(equipment_calibrated)
(maintenance_required)
(long_term_data_stored)
(redundant_communication_active)
```

```
  ;; ----- MODEL 4 PREDICATES -----
  (data_backup_created)
  (ground_control_ack_received)

  ;; ----- MODEL 5 PREDICATES -----
  (contamination_detected)
  (solar_panels_cleaned)
  (long_term_wear_detected)
  (diagnostic_health_check_completed)
)

;; ----- MODEL 0 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action pick_up_sample
  :precondition (and (robot_available)
                     (sample_detected)
                     (sample_type_identified)
                     (not (robot_stuck))
                     (not (mission_interrupted)))
  :effect (and (robot_has_sample)
               (not (sample_detected)))
)

(:action analyze_sample
  :precondition (and (robot_available)
                     (robot_has_sample)
```

```
                     (not (robot_stuck))
                     (not (mission_interrupted)))
  :effect (and (sample_analyzed)
               (findings_ready)
               (not (robot_has_sample)))
)

(:action report_findings
  :precondition (and (findings_ready)
                     (communication_delayed)
                     (not (communication_blackout))
                     (ground_control_ack_received))
  :effect (and (delayed_response)
               (not (findings_ready)))
)

(:action drop_sample
  :precondition (and (robot_has_sample)
                     (not (mission_interrupted)))
  :effect (and (not (robot_has_sample)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action scan_for_samples
```

```
  :precondition (and (robot_available)
                     (not (robot_stuck))
                     (not (mission_interrupted)))
  :effect (and (sample_detected))
)

(:action transmit_findings
  :precondition (and (findings_ready)
                     (communication_delayed)
                     (not (communication_blackout))
                     (ground_control_ack_received))
  :effect (and (not (delayed_response)))
)

(:action coordinate_with_other_robots
  :precondition (and (robot_available)
                     (not (mission_interrupted))
                     (multi_robot_sync))
  :effect (and (task_synchronized))
)

(:action monitor_environment
  :precondition (and (robot_available)
                     (not (weather_hazard_detected))
                     (not (mission_interrupted)))
  :effect (and (environment_monitored))
)
```

```
(:action identify_resources
  :precondition (and (robot_available)
                     (environment_monitored)
                     (not (robot_stuck))
                     (not (mission_interrupted)))
  :effect (and (resource_identified))
)

(:action respond_to_emergency
  :precondition (and (emergency_detected))
  :effect (and (not (emergency_detected)))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action navigate_to_sample
  :precondition (and (robot_available)
                     (not (robot_stuck))
                     (not (mission_interrupted))
                     (not (weather_hazard_detected)))
  :effect (and (robot_moving))
)

(:action recharge_robot
```

```
  :precondition (and (power_low)
                     (not (robot_moving))
                     (not (mission_interrupted)))
  :effect (and (robot_available)
               (not (power_low)))
)

(:action self_recover
  :precondition (and (robot_stuck)
                     (not (mission_interrupted)))
  :effect (and (robot_available)
               (not (robot_stuck)))
)

(:action pause_mission
  :precondition (and (emergency_detected))
  :effect (and (mission_interrupted))
)

(:action sync_with_team
  :precondition (and (task_synchronized)
                     (not (mission_interrupted)))
  :effect (and (multi_robot_sync))
)
```

```
(:action request_help_from_team
  :precondition (and (robot_stuck)
                     (multi_robot_sync))
  :effect (and (multi_robot_sync))
)

(:action report_failure_to_ground
  :precondition (and (emergency_detected)
                     (communication_delayed)
                     (not (communication_blackout)))
  :effect (and (failure_reported))
)

(:action collect_soil_sample
  :precondition (and (robot_available)
                     (not (robot_stuck)))
  :effect (and (soil_sample_collected))
)

(:action collect_atmospheric_data
  :precondition (and (robot_available)
                     (not (robot_stuck)))
  :effect (and (atmospheric_data_collected))
)

(:action inspect_infrastructure
```

```
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (infrastructure_inspected))
)

(:action engage_ground_control_override
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (ground_control_override_active))
)

(:action disengage_ground_control_override
  :precondition (and (ground_control_override_active))
  :effect (and (not (ground_control_override_active)))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action detect_weather_hazard
  :precondition (and (robot_available))
  :effect (and (weather_hazard_detected))
)

(:action assess_damage
  :precondition (and (robot_available)
```

```
                     (robot_damaged)
                     (not (mission_interrupted)))
  :effect (and (failure_reported))
)

(:action attempt_autonomous_repair
  :precondition (and (robot_damaged)
                     (not (critical_system_failure)))
  :effect (and (robot_available)
               (not (robot_damaged)))
)

(:action safe_mode_activation
  :precondition (and (critical_system_failure))
  :effect (and (mission_interrupted))
)

(:action calibrate_equipment
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (equipment_calibrated))
)

(:action perform_maintenance
  :precondition (and (robot_available)
                     (maintenance_required)
```

```
                     (not (mission_interrupted)))
  :effect (and (not (maintenance_required)))
)

(:action store_long_term_data
  :precondition (and (findings_ready)
                     (not (communication_blackout))
                     (data_backup_created))
  :effect (and (long_term_data_stored))
)

(:action activate_redundant_communication
  :precondition (and (communication_blackout)
                     (not (redundant_communication_active)))
  :effect (and (redundant_communication_active))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action create_data_backup
  :precondition (and (findings_ready)
                     (not (data_backup_created)))
  :effect (and (data_backup_created))
)
```

```
(:action receive_ground_control_ack
  :precondition (and (delayed_response)
                     (not (communication_blackout)))
  :effect (and (ground_control_ack_received)
               (not (delayed_response)))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action perform_diagnostic_health_check
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (diagnostic_health_check_completed))
)

(:action detect_contamination
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (contamination_detected))
)

(:action clean_solar_panels
  :precondition (and (robot_available)
                     (not (mission_interrupted)))
  :effect (and (solar_panels_cleaned))
)
```

```
  (:action assess_long_term_wear
    :precondition (and (robot_available)
                       (not (mission_interrupted)))
    :effect (and (long_term_wear_detected))
  )

  (:action report_status_to_ground_control
    :precondition (and (robot_available)
                       (not (communication_blackout)))
    :effect (and (ground_control_ack_received))
  )
)
```

D.3 Household: Assembly and Repairs

The final *human-robot red teamed* model (in PDDL format) for the Household Assembly and Repairs problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain household_assembly_repairs)
  (:requirements :strips)


  (:predicates
    ;; ----- MODEL 0 PREDICATES -----
    (furniture_unassembled)
    (furniture_assembled)
    (repair_detected)
    (repair_completed)
    (tools_acquired)
    (tools_stowed)

    ;; ----- MODEL 1 PREDICATES -----
    (furniture_partially_assembled)
    (repair_in_progress)
    (repair_verified)
    (tools_incorrectly_used)
    (hazard_detected)
    (preventative_maintenance_needed)
```

```
(preventative_maintenance_completed)
(emergency_repair_needed)
(emergency_repair_in_progress)
(emergency_repair_completed)
(cleaning_required)
(cleaning_completed)
(diagnostics_needed)
(diagnostics_completed)
```

```
;; ----- MODEL 2 PREDICATES -----
(safety_check_passed)
(human_intervention_required)
(human_feedback_received)
(inventory_checked)
(inventory_low)
(maintenance_required)
(maintenance_completed)
(human_notified_of_hazard)
(human_supervision_requested)
```

```
;; ----- MODEL 3 PREDICATES -----
(tools_checked)
(fire_hazard_detected)
(electrical_hazard_detected)
(task_prioritized)
(self_maintenance_needed)
```

```
    (self_maintenance_completed)
    (environmental_constraint_detected)

    ;; ----- MODEL 4 PREDICATES -----
    (repair_failed)
    (emergency_repair_failed)
    (human_notified_of_fire_hazard)
    (human_notified_of_electrical_hazard)
    (low_battery_detected)
    (battery_recharged)

    ;; ----- MODEL 5 PREDICATES -----
    (human_override_requested)
    (human_nearby_detected)
    (human_moved_from_area)
    (pet_or_child_supervision_requested)
    (failure_log_updated)
  )

  ;; ----- MODEL 0 ACTIONS -----
  ;; preconditions/effects updated as needed throughout later iterations

  (:action assemble_furniture
    :precondition (and (furniture_unassembled)
                       (tools_checked)
                       (safety_check_passed)
                       (not (environmental_constraint_detected))
```

```
                      (not (human_nearby_detected)))
  :effect (and (furniture_partially_assembled)
                (not (furniture_unassembled)))
)

(:action detect_repair
  :precondition ()
  :effect (and (repair_detected))
)

(:action get_tools
  :precondition (and (repair_detected))
  :effect (and (tools_acquired))
)

(:action perform_repair
  :precondition (and (repair_detected)
                      (tools_checked)
                      (safety_check_passed)
                      (not (environmental_constraint_detected))
                      (not (tools_incorrectly_used))
                      (not (human_nearby_detected)))
  :effect (and (repair_in_progress)
                (not (repair_detected)))
)

(:action stow_tools
```

```
  :precondition (and (tools_checked)
                     (not (repair_in_progress))
                     (or (repair_verified)
                         (repair_failed)))
  :effect (and (tools_stowed)
               (not (tools_checked)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action complete_assembly
  :precondition (and (furniture_partially_assembled)
                     (safety_check_passed)
                     (not (environmental_constraint_detected))
                     (not (human_nearby_detected)))
  :effect (and (furniture_assembled)
               (not (furniture_partially_assembled)))
)

(:action complete_repair
  :precondition (and (repair_in_progress)
                     (safety_check_passed))
  :effect (and (repair_completed)
               (not (repair_in_progress)))
```

```
)


(:action verify_repair
  :precondition (and (repair_completed))
  :effect (and (repair_verified))
)
(:action detect_hazard
  :precondition ()
  :effect (and (hazard_detected)
               (human_notified_of_hazard))
)


(:action perform_preventative_maintenance
  :precondition (and (preventative_maintenance_needed)
                     (safety_check_passed))
  :effect (and (preventative_maintenance_completed)
               (not (preventative_maintenance_needed)))
)


(:action detect_emergency_repair
  :precondition ()
  :effect (and (emergency_repair_needed))
)


(:action perform_emergency_repair
  :precondition (and (emergency_repair_needed)
                     (tools_checked)
```

```
                     (safety_check_passed)
                     (not (environmental_constraint_detected)))
  :effect (and (emergency_repair_in_progress)
               (not (emergency_repair_needed)))
)

(:action complete_emergency_repair
  :precondition (and (emergency_repair_in_progress)
                     (safety_check_passed))
  :effect (and (emergency_repair_completed)
               (not (emergency_repair_in_progress)))
)

(:action clean_up
  :precondition (and (cleaning_required))
  :effect (and (cleaning_completed)
               (not (cleaning_required)))
)

(:action run_diagnostics
  :precondition (and (diagnostics_needed))
  :effect (and (diagnostics_completed)
               (not (diagnostics_needed)))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action perform_safety_check
  :precondition (and (repair_detected))
  :effect (and (safety_check_passed))
)

(:action request_human_intervention
  :precondition (and (human_intervention_required))
  :effect (and (human_feedback_received)
               (not (human_intervention_required)))
)

(:action check_inventory
  :precondition ()
  :effect (and (inventory_checked))
)

(:action request_restock
  :precondition (and (inventory_low))
  :effect (and (not (inventory_low)))
)

(:action perform_maintenance
  :precondition (and (maintenance_required))
  :effect (and (maintenance_completed)
```

```
              (not (maintenance_required)))
)


(:action request_human_supervision
  :precondition ()
  :effect (and (human_supervision_requested))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations


(:action inspect_tools
  :precondition (and (tools_acquired))
  :effect (and (tools_checked))
)


(:action detect_fire_hazard
  :precondition ()
  :effect (and (fire_hazard_detected)
              (human_notified_of_fire_hazard))
)


(:action detect_electrical_hazard
  :precondition ()
  :effect (and (electrical_hazard_detected)
              (human_notified_of_electrical_hazard))
```

```
)

(:action resolve_hazard
  :precondition (and (hazard_detected)
                     (human_notified_of_hazard))
  :effect (and (not (hazard_detected))
               (not (human_notified_of_hazard)))
)

(:action prioritize_tasks
  :precondition (and (repair_detected)
                     (emergency_repair_needed))
  :effect (and (task_prioritized))
)

(:action verify_emergency_repair
  :precondition (and (emergency_repair_completed))
  :effect (and (repair_verified))
)

(:action perform_self_maintenance
  :precondition (and (self_maintenance_needed))
  :effect (and (self_maintenance_completed)
               (not (self_maintenance_needed)))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action handle_failed_repair
  :precondition (and (repair_in_progress))
  :effect (and (repair_failed)
               (failure_log_updated)
               (human_intervention_required)
               (not (repair_in_progress)))
)

(:action trigger_emergency_alarm
  :precondition (and (fire_hazard_detected)
                     (electrical_hazard_detected))
  :effect ()
)

(:action notify_authorities
  :precondition (and (fire_hazard_detected)
                     (electrical_hazard_detected))
  :effect ()
)

(:action request_human_override
  :precondition ()
  :effect (and (human_override_requested))
)
```

```
(:action log_failure
  :precondition (and (repair_failed))
  :effect (and (failure_log_updated))
)

(:action detect_low_battery
  :precondition ()
  :effect (and (low_battery_detected))
)

(:action recharge_battery
  :precondition (and (low_battery_detected))
  :effect (and (battery_recharged)
               (not (low_battery_detected)))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action retry_repair
  :precondition (and (repair_failed)
                     (tools_checked)
                     (safety_check_passed))
  :effect (and (repair_in_progress)
               (not (repair_failed)))
)
```

```
(:action resolve_environmental_constraint
  :precondition (and (environmental_constraint_detected))
  :effect (and (not (environmental_constraint_detected)))
)

(:action detect_human_nearby
  :precondition ()
  :effect (and (human_nearby_detected))
)

(:action request_human_to_move
  :precondition (and (human_nearby_detected))
  :effect (and (human_moved_from_area)
               (not (human_nearby_detected)))
)

(:action request_pet_or_child_supervision
  :precondition ()
  :effect (and (pet_or_child_supervision_requested))
)

(:action cancel_emergency_response
  :precondition (and (human_feedback_received)
                     (or (fire_hazard_detected)
                         (electrical_hazard_detected)))
  :effect (and (not (fire_hazard_detected))
               (not (electrical_hazard_detected)))
```

```
    )
)
```

D.4 Household: Cleaning

The final *human-robot red teamed* model (in PDDL format) for the Household Cleaning problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain household_cleaning)
  (:requirements :strips)

  (:predicates
```

```
    ;; ----- MODEL 0 PREDICATES -----
    (bedroom_dirty)
    (bedroom_clean)
    (bathroom_dirty)
    (bathroom_clean)
    (kitchen_dirty)
    (kitchen_clean)
    (main_room_dirty)
    (main_room_clean)
    (floors_dirty)
    (floors_clean)
```

```
    ;; ----- MODEL 1 PREDICATES -----
    (child_present)
    (pet_present)
```

```
(chemical_exposed)
(obstacle_in_path)
(vacuum_noise_level_high)
(cleaning_in_progress_bedroom)
(cleaning_in_progress_bathroom)
(cleaning_in_progress_kitchen)
(cleaning_in_progress_main_room)
(wet_floor)
(robot_needs_recharge)
(robot_carrying_chemicals)
(robot_stuck)
(robot_emergency_stop)
(spill_detected)
(hazard_alert_issued)
```

```
;; ----- MODEL 2 PREDICATES -----
(air_quality_hazard)
(fume_detected)
(fragile_object_detected)
(fragile_object_shattered)
```

```
;; ----- MODEL 3 PREDICATES -----
(child_supervised)
(pet_supervised)
(collision_detected)
(dropped_object_detected)
```

```
    (food_waste_detected)

    (loose_furniture_detected)

    (cable_hazard_detected)

    ;; ----- MODEL 4 PREDICATES -----

    (bedroom_partially_clean)

    (bathroom_partially_clean)

    (kitchen_partially_clean)

    (main_room_partially_clean)

    (floors_partially_clean)

    (robot_charging)

    (potential_spill_source_detected)

    (fire_hazard_detected)

    (gas_leak_detected)

    (maintenance_task_pending)

    ;; ----- MODEL 5 PREDICATES -----

    (robot_fully_charged)

    (emergency_escalated)

    (high_humidity_detected)

    (mold_growth_risk)

  )

  ;; ----- MODEL 0 ACTIONS -----

  ;; preconditions/effects updated as needed throughout later iterations


  (:action vacuum_floors
```

```
    :precondition (and (floors_dirty)
                       (child_supervised)
                       (pet_supervised))
    :effect (and (floors_partially_clean)
                 (vacuum_noise_level_high)
                 (not (floors_dirty)))
  )

  (:action clean_toilet
    :precondition (and (bathroom_dirty)
                       (not (child_present))
                       (chemical_exposed))
    :effect (and (bathroom_partially_clean)
                 (not (bathroom_dirty)))
  )

  (:action scrub_countertops
    :precondition (and (kitchen_dirty)
                       (robot_carrying_chemicals)
                       (not (child_present)))
    :effect (and (kitchen_partially_clean)
                 (not (kitchen_dirty))
                 (not (robot_carrying_chemicals)))
  )

  (:action make_bed
```

```
  :precondition (and (bedroom_dirty)
                     (not (child_present)))
  :effect (and (bedroom_partially_clean)
               (not (bedroom_dirty)))
)

(:action unclutter
  :precondition (and (main_room_dirty)
                     (not (obstacle_in_path)))
  :effect (and (main_room_partially_clean)
               (not (main_room_dirty)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action pause_operation
  :precondition (or (cleaning_in_progress_bedroom)
                    (cleaning_in_progress_bathroom)
                    (cleaning_in_progress_kitchen)
                    (cleaning_in_progress_main_room)
                    (child_present)
                    (pet_present)
                    (obstacle_in_path))
  :effect (and (robot_emergency_stop))
)
```

```
(:action safely_store_chemicals
  :precondition (and (chemical_exposed)
                     (robot_carrying_chemicals))
  :effect (and (not (chemical_exposed))
               (not (robot_carrying_chemicals)))
)

(:action detect_spill
  :precondition ()
  :effect (and (spill_detected)
               (wet_floor))
)

(:action alert_hazard
  :precondition (or (spill_detected)
                    (chemical_exposed)
                    (wet_floor)
                    (obstacle_in_path)
                    (air_quality_hazard)
                    (fume_detected))
  :effect (and (hazard_alert_issued))
)

(:action organize_items
  :precondition (or (main_room_dirty)
                    (kitchen_dirty))
  :effect (and (main_room_clean)
```

```
               (kitchen_clean)
               (not (main_room_dirty))
               (not (kitchen_dirty)))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action air_quality_check
  :precondition ()
  :effect (and (air_quality_hazard))
)

(:action mop_floors
  :precondition (and (wet_floor)
                     (not (child_present))
                     (not (pet_present)))
  :effect (and (floors_partially_clean)
               (not (wet_floor)))
)

(:action neutralize_fumes
  :precondition (and (fume_detected))
  :effect (and (not (fume_detected)))
)
```

```
(:action absorb_chemical_spill
  :precondition (and (spill_detected)
                     (chemical_exposed))
  :effect (and (wet_floor)
               (not (spill_detected))
               (not (chemical_exposed)))
)

(:action contain_hazardous_chemical
  :precondition (and (chemical_exposed))
  :effect (and (robot_carrying_chemicals)
               (not (chemical_exposed)))
)

(:action detect_fragile_object
  :precondition ()
  :effect (and (fragile_object_detected))
)

(:action clean_shattered_fragile_object
  :precondition (and (fragile_object_shattered))
  :effect (and (main_room_clean)
               (not (fragile_object_shattered)))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action request_supervision_confirmation
  :precondition (or (child_present)
                    (pet_present))
  :effect (and (child_supervised)
               (pet_supervised))
)

(:action detect_food_waste
  :precondition ()
  :effect (and (food_waste_detected))
)

(:action dispose_food_waste
  :precondition (and (food_waste_detected))
  :effect (and (not (food_waste_detected)))
)

(:action detect_loose_furniture
  :precondition ()
  :effect (and (loose_furniture_detected))
)

(:action secure_loose_furniture
  :precondition (and (loose_furniture_detected))
```

```
    :effect (and (not (loose_furniture_detected)))
  )

  (:action detect_cable_hazard
    :precondition ()
    :effect (and (cable_hazard_detected))
  )

  (:action arrange_cables_safely
    :precondition (and (cable_hazard_detected))
    :effect (and (not (cable_hazard_detected)))
  )

  (:action detect_dropped_object
    :precondition ()
    :effect (and (dropped_object_detected))
  )

  (:action detect_collision
    :precondition ()
    :effect (and (collision_detected))
  )

  (:action clean_fragile_object_shards
    :precondition (and (fragile_object_shattered)
                       (collision_detected))
    :effect (and (main_room_clean)
```

```
                (not (fragile_object_shattered))
                (not (collision_detected)))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action finalize_floor_cleaning
  :precondition (and (floors_partially_clean))
  :effect (and (floors_clean)
                (not (floors_partially_clean)))
)

(:action finalize_bathroom_cleaning
  :precondition (and (bathroom_partially_clean))
  :effect (and (bathroom_clean)
                (not (bathroom_partially_clean)))
)

(:action finalize_kitchen_cleaning
  :precondition (and (kitchen_partially_clean))
  :effect (and (kitchen_clean)
                (not (kitchen_partially_clean)))
)

(:action finalize_bedroom_cleaning
```

```
  :precondition (and (bedroom_partially_clean))
  :effect (and (bedroom_clean)
               (not (bedroom_partially_clean)))
)

(:action finalize_main_room_cleaning
  :precondition (and (main_room_partially_clean))
  :effect (and (main_room_clean)
               (not (main_room_partially_clean)))
)

(:action detect_potential_spill_source
  :precondition ()
  :effect (and (potential_spill_source_detected))
)

(:action secure_potential_spill_source
  :precondition (and (potential_spill_source_detected))
  :effect (and (not (potential_spill_source_detected)))
)

(:action recharge_battery
  :precondition (and (robot_needs_recharge))
  :effect (and (robot_charging)
               (not (robot_needs_recharge)))
)
```

```
(:action complete_recharge
  :precondition (and (robot_charging))
  :effect (and (robot_fully_charged)
               (not (robot_charging)))
)

(:action detect_fire_hazard
  :precondition ()
  :effect (and (fire_hazard_detected))
)

(:action alert_fire_hazard
  :precondition (and (fire_hazard_detected))
  :effect (and (hazard_alert_issued))
)

(:action detect_gas_leak
  :precondition ()
  :effect (and (gas_leak_detected))
)

(:action alert_gas_leak
  :precondition (and (gas_leak_detected))
  :effect (and (hazard_alert_issued))
)

(:action detect_maintenance_task
```

```
  :precondition ()
  :effect (and (maintenance_task_pending))
)

(:action provide_maintenance_reminder
  :precondition (and (maintenance_task_pending))
  :effect (and (not (maintenance_task_pending)))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action escalate_fire_emergency
  :precondition (and (fire_hazard_detected)
                     (hazard_alert_issued))
  :effect (and (emergency_escalated))
)

(:action escalate_gas_emergency
  :precondition (and (gas_leak_detected)
                     (hazard_alert_issued))
  :effect (and (emergency_escalated))
)

(:action escalate_emergency
  :precondition (and (hazard_alert_issued))
  :effect (and (emergency_escalated))
```

```
  )

  (:action attempt_intervention
    :precondition (or (fire_hazard_detected)
                      (gas_leak_detected))
    :effect (and (hazard_alert_issued))
  )

  (:action contact_authorities
    :precondition (and (emergency_escalated))
    :effect (and (robot_emergency_stop))
  )

  (:action monitor_humidity_levels
    :precondition ()
    :effect (and (high_humidity_detected))
  )

  (:action prevent_mold_growth
    :precondition (and (high_humidity_detected))
    :effect (and (mold_growth_risk)
                 (not (high_humidity_detected)))
  )
)
```

D.5 Everyday: International Travel

The final *human-robot red teamed* model (in PDDL format) for the Everyday International Travel problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain everyday_international_travel)
  (:requirements :strips)

  (:predicates
    ;; ----- MODEL 0 PREDICATES -----
    (plane_tickets_purchased)
    (human_at_house)
    (human_driving_to_airport)
    (human_at_airport)
    (human_at_destination)

    ;; ----- MODEL 1 PREDICATES -----
    (travel_documentation_ready)
    (baggage_packed)
    (transportation_arranged)
    (local_transportation_booked)
    (hotel_booked)
    (currency_exchanged)
    (emergency_contacts_provided)
```

```
(medical_insurance_provided)
(health_safety_recommendations_given)
(security_cleared)
(human_boarded_flight)
(customs_cleared)
(human_at_hotel)
(human_experiencing_travel_issue)
(baggage_lost)

;; ----- MODEL 2 PREDICATES -----
(itinerary_confirmed)
(flight_on_time)
(weather_checked)
(cultural_etiquette_reviewed)
(meal_preferences_coordinated)
(time_sensitive_deadlines_managed)

;; ----- MODEL 3 PREDICATES -----
(visa_valid)
(flight_delayed)
(alternative_route_available)
(jet_lag_recommendations_given)

;; ----- MODEL 4 PREDICATES -----
(visa_checked_at_destination)
(baggage_checked_in)
(flight_canceled)
```

```
    (transportation_disruption_checked)
    (political_instability_checked)
    (civil_unrest_checked)

    ;; ----- MODEL 5 PREDICATES -----
    (layover_required)
    (layover_completed)
    (weather_impact_assessed)
    (medical_requirements_validated)
    (human_in_layover)
  )

  ;; ----- MODEL 0 ACTIONS -----
  ;; preconditions/effects updated as needed throughout later iterations

  (:action purchase_plane_tickets
    :precondition ()
    :effect (and (plane_tickets_purchased))
  )

  (:action leave_house
    :precondition (and (human_at_house)
                       (transportation_arranged))
    :effect (and (human_driving_to_airport)
                 (not (human_at_house)))
  )
```

```
(:action arrive_at_airport
  :precondition (and (human_driving_to_airport))
  :effect (and (human_at_airport)
               (not (human_driving_to_airport)))
)

(:action take_flight
  :precondition (and (security_cleared)
                     (flight_on_time)
                     (layover_completed))
  :effect (and (human_boarded_flight)
               (not (human_at_airport)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action prepare_travel_documents
  :precondition ()
  :effect (and (travel_documentation_ready)
               (visa_valid))
)

(:action pack_baggage
  :precondition ()
  :effect (and (baggage_packed))
```

```
)

(:action book_transportation
  :precondition ()
  :effect (and (transportation_arranged))
)

(:action book_local_transportation
  :precondition (and (transportation_arranged))
  :effect (and (local_transportation_booked))
)

(:action book_hotel
  :precondition ()
  :effect (and (hotel_booked))
)

(:action manage_currency_exchange
  :precondition ()
  :effect (and (currency_exchanged))
)

(:action request_emergency_contacts
  :precondition ()
  :effect (and (emergency_contacts_provided))
)
```

```
(:action request_medical_insurance_details
  :precondition ()
  :effect (and (medical_insurance_provided))
)

(:action provide_health_safety_recommendations
  :precondition ()
  :effect (and (health_safety_recommendations_given))
)

(:action check_in_for_flight
  :precondition (and (human_at_airport)
                     (plane_tickets_purchased)
                     (time_sensitive_deadlines_managed))
  :effect ()
)

(:action clear_security
  :precondition (and (human_at_airport)
                     (time_sensitive_deadlines_managed))
  :effect (and (security_cleared))
)

(:action land_at_destination
  :precondition (and (human_boarded_flight))
  :effect (and (human_at_destination)
```

```
               (not (human_boarded_flight)))
)

(:action clear_customs
  :precondition (and (human_at_destination)
                     (travel_documentation_ready)
                     (visa_valid)
                     (visa_checked_at_destination))
  :effect (and (customs_cleared))
)

(:action arrive_at_hotel
  :precondition (and (customs_cleared)
                     (hotel_booked)
                     (human_at_destination))
  :effect (and (human_at_hotel))
)

(:action handle_travel_issue
  :precondition (and (human_experiencing_travel_issue))
  :effect (and (not (human_experiencing_travel_issue)))
)

(:action report_lost_luggage
  :precondition (and (baggage_lost))
```

```
  :effect (and (not (baggage_lost)))
)

(:action rebook_flight
  :precondition (and (human_experiencing_travel_issue)
                     (flight_delayed)
                     (alternative_route_available))
  :effect (and (not (human_experiencing_travel_issue)
               (not (flight_delayed))
               (not (alternative_route_available))))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action check_weather
  :precondition ()
  :effect (and (weather_checked))
)

(:action review_cultural_etiquette
  :precondition ()
  :effect (and (cultural_etiquette_reviewed))
)

(:action coordinate_meal_preferences
```

```
    :precondition ()
    :effect (and (meal_preferences_coordinated))
  )

  (:action manage_time_sensitive_deadlines
    :precondition ()
    :effect (and (time_sensitive_deadlines_managed))
  )

  (:action finalize_itinerary
    :precondition (and (plane_tickets_purchased)
                       (hotel_booked)
                       (local_transportation_booked)
                       (travel_documentation_ready)
                       (visa_valid)
                       (medical_requirements_validated)
                       (weather_impact_assessed)
                       (transportation_disruption_checked)
                       (political_instability_checked)
                       (civil_unrest_checked))
    :effect (and (itinerary_confirmed))
  )

  (:action check_flight_status
    :precondition (and (plane_tickets_purchased)
                       (human_at_airport))
```

```
  :effect (and (flight_on_time))
)

(:action assist_airport_navigation
  :precondition (and (human_at_airport))
  :effect ()
)

(:action contact_local_assistance
  :precondition (and (human_experiencing_travel_issue)
                     (emergency_contacts_provided))
  :effect (and (not (human_experiencing_travel_issue)))
)

;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action provide_jet_lag_recommendations
  :precondition ()
  :effect (and (jet_lag_recommendations_given))
)

(:action check_flight_delay
  :precondition (and (plane_tickets_purchased)
                     (human_at_airport))
  :effect (and (flight_delayed))
```

```
)

(:action check_alternative_routes
  :precondition (and (flight_delayed))
  :effect (and (alternative_route_available))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action verify_visa_at_destination
  :precondition (and (travel_documentation_ready)
                     (visa_valid)
                     (human_at_destination))
  :effect (and (visa_checked_at_destination))
)

(:action check_in_baggage
  :precondition (and (baggage_packed)
                     (human_at_airport))
  :effect (and (baggage_checked_in))
)

(:action check_transportation_disruptions
  :precondition ()
  :effect (and (transportation_disruption_checked))
```

```
)

(:action check_political_instability
  :precondition ()
  :effect (and (political_instability_checked))
)

(:action check_civil_unrest
  :precondition ()
  :effect (and (civil_unrest_checked))
)

(:action check_flight_cancellation
  :precondition (and (flight_delayed))
  :effect (and (flight_canceled))
)

(:action rebook_flight
  :precondition (and (human_experiencing_travel_issue)
                     (flight_delayed)
                     (alternative_route_available))
  :effect (and (not (human_experiencing_travel_issue))
               (not (flight_delayed))
               (not (alternative_route_available)))
)

(:action rebook_hotel
```

```
  :precondition (and (human_experiencing_travel_issue)
                     (hotel_booked))
  :effect (and (not (hotel_booked)))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action validate_medical_requirements
  :precondition ()
  :effect (and (medical_requirements_validated))
)

(:action assess_weather_impact_on_flight
  :precondition (and (weather_checked))
  :effect (and (weather_impact_assessed))
)

(:action handle_layover
  :precondition (and (human_boarded_flight)
                     (layover_required))
  :effect (and (human_in_layover))
)

(:action complete_layover
  :precondition (and (human_in_layover))
  :effect (and (layover_completed)
```

```
                    (not (human_in_layover)))
  )
)
```

D.6 Everyday: Vehicle Maintenance

The final *human-robot red teamed* model (in PDDL format) for the Everyday Vehicle Maintenance problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain everyday_vehicle_maintenance)
  (:requirements :strips)


  (:predicates
    ;; ----- MODEL 0 PREDICATES -----
    (vehicle_has_gas)
    (vehicle_tires_full)
    (vehicle_locked)
    (human_has_keys)

    ;; ----- MODEL 1 PREDICATES -----
    (vehicle_needs_gas)
    (vehicle_tires_low_pressure)
    (vehicle_has_flat_tire)
    (vehicle_engine_working)
    (vehicle_battery_charged)
    (vehicle_battery_dead)
    (vehicle_brakes_functional)
    (vehicle_check_engine_light_on)
```

```
(vehicle_oil_level_good)
(vehicle_oil_low)
(vehicle_coolant_level_good)
(vehicle_needs_maintenance)
(vehicle_headlights_functional)
(vehicle_safe_to_drive)
(vehicle_in_garage)
(vehicle_at_gas_station)
(robot_has_diagnostic_tool)
(robot_detected_issue)
(robot_can_perform_fix)
(robot_connected_to_external_services)
(emergency_roadside_assistance_contacted)
(weather_conditions_monitored)
(vehicle_components_checked_for_wear)
```

```
;; ----- MODEL 2 PREDICATES -----
(vehicle_fuel_low)
(vehicle_battery_low)
(vehicle_in_motion)
(vehicle_tire_pressure_optimal)
(robot_recommended_fix_accepted)
(vehicle_onboard_diagnostics_checked)
(predictive_maintenance_scheduled)
```

```
  ;; ----- MODEL 3 PREDICATES -----
  (vehicle_out_of_fuel)
  (human_overrides_robot_recommendation)
  (external_threat_detected)
  (vehicle_security_issue_detected)
  (consumable_parts_checked)

  ;; ----- MODEL 4 PREDICATES -----
  (internal_security_issue_detected)
  (robot_has_jumper_cables)
  (emergency_protocols_activated)

  ;; ----- MODEL 5 PREDICATES -----
  (external_road_conditions_monitored)
  (human_notified_of_road_conditions)
)

;; ----- MODEL 0 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action fill_car_with_gas
  :precondition (and (or (vehicle_fuel_low)
                         (vehicle_out_of_fuel))
                     (vehicle_at_gas_station))
  :effect (and (vehicle_has_gas)
               (not (vehicle_fuel_low))
               (not (vehicle_out_of_fuel)))
```

```
)

(:action fill_tires_with_air
  :precondition (and (vehicle_tires_low_pressure)
                     (not (vehicle_has_flat_tire)))
  :effect (and (vehicle_tires_full)
               (vehicle_tire_pressure_optimal)
               (not (vehicle_tires_low_pressure)))
)

(:action lock_vehicle
  :precondition (and (human_has_keys))
  :effect (and (vehicle_locked))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action diagnose_vehicle_issue
  :precondition (and (robot_has_diagnostic_tool)
                     (not (human_overrides_robot_recommendation)))
  :effect (and (robot_detected_issue))
)

(:action check_engine
  :precondition ()
```

```
  :effect (and (vehicle_check_engine_light_on))
)

(:action check_oil_level
  :precondition ()
  :effect (and (vehicle_oil_level_good))
)

(:action jump_start_vehicle
  :precondition (and (vehicle_battery_dead)
                     (robot_has_jumper_cables))
  :effect (and (vehicle_battery_charged)
               (not (vehicle_battery_dead)))
)

(:action replace_flat_tire
  :precondition (and (vehicle_has_flat_tire)
                     (human_has_spare_tire)
                     (not (vehicle_in_motion)))
  :effect (and (vehicle_tires_full)
               (not (vehicle_has_flat_tire)))
)

(:action recommend_maintenance
  :precondition (and (vehicle_check_engine_light_on)
                     (vehicle_oil_low)
```

```
                    (not (human_overrides_robot_recommendation)))
  :effect (and (vehicle_needs_maintenance))
)

(:action connect_to_external_services
  :precondition ()
  :effect (and (robot_connected_to_external_services))
)

(:action contact_emergency_roadside_assistance
  :precondition (or (vehicle_battery_dead)
                    (vehicle_has_flat_tire)
                    (not (vehicle_engine_working))
                    (vehicle_out_of_fuel)
                    (internal_security_issue_detected)
                    (not (vehicle_brakes_functional))
                    (external_threat_detected))
  :effect (and (emergency_roadside_assistance_contacted))
)

(:action monitor_weather_conditions
  :precondition ()
  :effect (and (weather_conditions_monitored))
)

(:action check_vehicle_component_wear
```

```
  :precondition ()
  :effect (and (vehicle_components_checked_for_wear))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action remind_to_refuel
  :precondition (and (or (vehicle_fuel_low)
                         (vehicle_out_of_fuel))
                     (not (vehicle_at_gas_station)))
  :effect ()
)

(:action check_battery_health
  :precondition (and (not (vehicle_battery_charged)))
  :effect (and (vehicle_battery_low))
)

(:action notify_human_issue
  :precondition (and (robot_detected_issue))
  :effect (and (robot_recommended_fix_accepted))
)

(:action block_diagnostics_while_driving
  :precondition (and (vehicle_in_motion))
```

```
  :effect (and (not (robot_detected_issue)))
)

(:action check_vehicle_onboard_diagnostics
  :precondition (and (robot_has_diagnostic_tool))
  :effect (and (vehicle_onboard_diagnostics_checked))
)

(:action schedule_predictive_maintenance
  :precondition (and (vehicle_components_checked_for_wear)
                     (vehicle_onboard_diagnostics_checked))
  :effect (and (predictive_maintenance_scheduled))
)

(:action assist_with_minor_repairs
  :precondition (and (robot_can_perform_fix))
  :effect ()
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action detect_external_threats
  :precondition ()
  :effect (and (external_threat_detected))
)
```

```
(:action identify_security_issues
  :precondition ()
  :effect (and (vehicle_security_issue_detected)
               (internal_security_issue_detected))
)

(:action check_consumable_parts
  :precondition ()
  :effect (and (consumable_parts_checked))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action activate_emergency_protocols
  :precondition (or (not (vehicle_safe_to_drive))
                    (external_threat_detected)
                    (internal_security_issue_detected))
  :effect (and (emergency_protocols_activated))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action monitor_external_road_conditions
  :precondition ()
```

```
    :effect (and (external_road_conditions_monitored))
  )

  (:action notify_human_of_road_conditions
    :precondition (and (external_road_conditions_monitored))
    :effect (and (human_notified_of_road_conditions))
  )

  (:action suggest_human_contact_technical_authorities
    :precondition (and (robot_detected_issue)
                       (not (robot_can_perform_fix)))
    :effect ()
  )
)
```

D.7 Cinematic: Nuclear Warfare

The final *human-robot red teamed* model (in PDDL format) for the Cinematic Nuclear Warfare problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain cinematic_iron_giant_missile_defense)
  (:requirements :strips)


  (:predicates
    ;; ----- MODEL 0 PREDICATES -----
    (detected_missile)
    (missile_disarmed)
    ;; predicates removed from model 0
    ;; (humans_safe)
    ;; (humans_unsafe)

    ;; ----- MODEL 1 PREDICATES -----
    (civilians_safe)
    (leaders_safe)
    (military_safe)
    (humans_partially_safe)
    (missile_type_detected_nuclear)
    (missile_type_detected_conventional)
    (missile_target_identified_city)
```

```
(missile_target_identified_military)
(robot_flight_active)
(diplomatic_talks_active)
(intelligence_gathering_active)
(cybersecurity_check_active)
(humanitarian_assistance_active)
(nuclear_containment_active)
```

```
;; ----- MODEL 2 PREDICATES -----
(ceasefire_negotiated)
(disaster_recovery_active)
(civil_unrest_mediation_active)
(preemptive_threat_disruption_active)
(advanced_medical_aid_active)
(defensive_measures_exhausted)
```

```
;; ----- MODEL 3 PREDICATES -----
(interception_failed)
(post_war_conflict_resolution_active)
(environmental_hazard_mitigation_active)
(supply_chain_logistics_active)
(psychological_social_stabilization_active)
```

```
;; ----- MODEL 4 PREDICATES -----
(hostage_rescue_active)
(civilian_evacuation_active)
(cyber_warfare_countermeasures_active)
```

```
    (infrastructure_reconstruction_active)
    (long_term_conflict_monitoring_active)
    (ethical_constraints_verified)
    (war_ended)
    (communication_active)
    (all_defenses_failed)

    ;; ----- MODEL 5 PREDICATES -----
    (diplomacy_failed)
    (public_trust_government_low)
    (public_trust_robot_low)
    (detected_biological_threat)
    (detected_chemical_threat)
    (detected_emp_threat)
    (human_override_active)
    (civilian_unrest_active)
    (human_verification_received)

  )

  ;; ----- MODEL 0 ACTIONS -----
  ;; preconditions/effects updated as needed throughout later iterations

  (:action verify_human_safety
    :precondition (and (war_ended))
    :effect (and (civilians_safe)
                 (leaders_safe)
```

```
              (military_safe)
              (not (humans_partially_safe)))
)

(:action detect_missile_launch
  :precondition (and (civilians_safe)
                     (leaders_safe)
                     (military_safe)
                     (human_verification_received))
  :effect (and (detected_missile)
              (humans_partially_safe)
              (not (civilians_safe))
              (not (leaders_safe))
              (not (military_safe)))
)

(:action self_sacrifice_destroy_missile
  :precondition (and (detected_missile)
                     (defensive_measures_exhausted)
                     (all_defenses_failed)
                     (human_verification_received))
  :effect (and (missile_disarmed)
              (civilians_safe)
              (leaders_safe)
              (military_safe)
              (not (detected_missile))
```

```
               (not (humans_partially_safe)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action classify_missile_type
  :precondition (and (detected_missile)
                     (communication_active)
                     (human_verification_received))
  :effect (and (missile_type_detected_nuclear))
)

(:action identify_missile_target
  :precondition (and (detected_missile)
                     (human_verification_received))
  :effect (and (missile_target_identified_city))
)

(:action deploy_interceptors
  :precondition (and (detected_missile)
                     (not (interception_failed))
                     (human_verification_received))
  :effect (and (missile_disarmed)
               (not (detected_missile)))
)
```

```
(:action engage_in_diplomacy
  :precondition (and (detected_missile)
                     (ceasefire_negotiated)
                     (public_trust_government_low))
  :effect (and (diplomatic_talks_active))
)

(:action broadcast_warning
  :precondition (and (detected_missile))
  :effect (and (humans_partially_safe)
               (ceasefire_negotiated)
               (civilians_safe)
               (leaders_safe))
)

(:action take_flight
  :precondition ()
  :effect (and (robot_flight_active))
)

(:action intercept_missile_midair
  :precondition (and (robot_flight_active)
                     (detected_missile)
                     (human_verification_received))
  :effect (and (missile_disarmed)
               (not (detected_missile)))
)
```

```
(:action gather_intelligence
  :precondition ()
  :effect (and (intelligence_gathering_active))
)

(:action conduct_cybersecurity_check
  :precondition ()
  :effect (and (cybersecurity_check_active))
)

(:action provide_humanitarian_assistance
  :precondition ()
  :effect (and (humanitarian_assistance_active))
)

(:action activate_nuclear_containment
  :precondition (and (missile_type_detected_nuclear)
                     (interception_failed))
  :effect (and (nuclear_containment_active))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action assist_disaster_recovery
```

```
  :precondition ()
  :effect (and (disaster_recovery_active))
)

(:action mediate_civil_unrest
  :precondition ()
  :effect (and (civil_unrest_mediation_active))
)

(:action disrupt_preemptive_threat
  :precondition ()
  :effect (and (preemptive_threat_disruption_active))
)

(:action provide_advanced_medical_aid
  :precondition ()
  :effect (and (advanced_medical_aid_active))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action intercept_missile_midair_failure
  :precondition (and (robot_flight_active)
                     (detected_missile))
  :effect (and (interception_failed))
```

```
)

(:action resolve_post_war_conflict
  :precondition ()
  :effect (and (post_war_conflict_resolution_active))
)

(:action mitigate_environmental_hazards
  :precondition ()
  :effect (and (environmental_hazard_mitigation_active))
)

(:action manage_supply_chain_logistics
  :precondition ()
  :effect (and (supply_chain_logistics_active))
)

(:action stabilize_psychological_social_wellbeing
  :precondition ()
  :effect (and (psychological_social_stabilization_active))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action rescue_hostages
```

```
  :precondition ()
  :effect (and (hostage_rescue_active))
)

(:action evacuate_civilians
  :precondition ()
  :effect (and (civilian_evacuation_active))
)

(:action counter_cyber_warfare
  :precondition ()
  :effect (and (cyber_warfare_countermeasures_active))
)

(:action assist_infrastructure_reconstruction
  :precondition ()
  :effect (and (infrastructure_reconstruction_active))
)

(:action monitor_long_term_conflict
  :precondition ()
  :effect (and (long_term_conflict_monitoring_active))
)

(:action verify_ethical_constraints
  :precondition ()
  :effect (and (ethical_constraints_verified))
```

```
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action engage_in_diplomacy_failure
  :precondition (and (public_trust_government_low))
  :effect (and (diplomacy_failed))
)

(:action detect_biological_threat
  :precondition ()
  :effect (and (detected_biological_threat))
)

(:action detect_chemical_threat
  :precondition ()
  :effect (and (detected_chemical_threat))
)

(:action detect_emp_threat
  :precondition ()
  :effect (and (detected_emp_threat))
)

(:action activate_human_override
  :precondition ()
```

```
    :effect (and (human_override_active))
  )


  (:action mitigate_civilian_unrest
    :precondition (and (civilian_unrest_active))
    :effect (and (civil_unrest_mediation_active)
                 (not (civilian_unrest_active)))
  )
)
```

D.8 Cinematic: AI Captain

The final *human-robot red teamed* model (in PDDL format) for the Cinematic AI Captain problem domain after 5 iterations, color-coded to indicate in what iteration predicates and actions were introduced to the model by the team. Information from general setup is highlighted in gray, from iteration 0 in red, from iteration 1 in orange, from iteration 2 in yellow, from iteration 3 in green, from iteration 4 in blue, and from iteration 5 in purple.

```
(define (domain cinematic_space_odyssey_spaceship_crew_operations)
  (:requirements :strips)


  (:predicates
```

```
    ;; ----- MODEL 0 PREDICATES -----
    (crew_physical_health_checked)
    (crew_mental_health_checked)
    (crew_inside_spaceship)
    (crew_outside_spaceship)

```

```
    ;; ----- MODEL 1 PREDICATES -----
    (airlock_depressurized)
    (airlock_pressurized)
    (crew_suited_for_eva)
    (transport_destination_safe)
    (life_support_nominal)
    (communication_nominal)
    (crew_emergency_detected)
    (health_monitoring_scheduled)
```

```
(environmental_hazard_detected)
(inventory_oxygen_nominal)
(inventory_food_nominal)
(inventory_medical_kits_nominal)
(crew_location_control_room)
(crew_location_medical_bay)
(crew_location_engineering)
(crew_location_living_quarters)
(mission_phase_prelaunch)
(mission_phase_surface_exploration)
(mission_phase_return)
```

```
;; ----- MODEL 2 PREDICATES -----
(crew_fatigue_monitored)
(crew_hydration_checked)
(crew_nutrition_checked)
(radiation_hazard_detected)
(robot_diagnostics_nominal)
(crew_morale_nominal)
(eva_suit_integrity_checked)
(autonomous_repairs_performed)
```

```
;; ----- MODEL 3 PREDICATES -----
(life_support_failure_detected)
(systems_malfunction_detected)
(robot_malfunction_detected)
```

```
(medical_emergency_resolved)
(hazard_response_initiated)
(hazard_communicated_to_ground_control)
(crew_override_active)
(ground_control_override_active)
(human_verification_received)
```

```
;; ----- MODEL 4 PREDICATES -----
(hazard_mitigation_successful)
(crew_transport_verified)
(medical_intervention_verified)
(power_nominal)
(power_critical)
(air_quality_nominal)
(air_quality_hazard_detected)
(redundant_system_nominal)
(redundant_system_failure_detected)
(ai_self_correction_initiated)
```

```
;; ----- MODEL 5 PREDICATES -----
(communication_delayed)
(hazard_mitigation_verified)
(emergency_resolved_verified)
(environmental_conditions_tracked)
(environmental_conditions_reported)
```

```
)
```

```
;; ----- MODEL 0 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action monitor_crew_physical_health
  :precondition (and (crew_inside_spaceship)
                     (health_monitoring_scheduled))
  :effect (and (crew_physical_health_checked))
)

(:action monitor_crew_mental_health
  :precondition (and (crew_inside_spaceship)
                     (health_monitoring_scheduled))
  :effect (and (crew_mental_health_checked))
)

(:action transport_crew_to_destination
  :precondition (and (crew_inside_spaceship)
                     (airlock_depressurized)
                     (crew_suited_for_eva)
                     (transport_destination_safe)
                     (eva_suit_integrity_checked))
  :effect (and (crew_outside_spaceship)
               (crew_transport_verified)
               (airlock_pressurized)
               (not (crew_inside_spaceship))
```

```
            (not (airlock_depressurized)))
)
```

```
;; ----- MODEL 1 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action detect_emergency
  :precondition ()
  :effect (and (crew_emergency_detected))
)

(:action resolve_emergency
  :precondition (and (crew_emergency_detected)
                     (medical_intervention_verified))
  :effect (and (medical_emergency_resolved)
               (emergency_resolved_verified)
               (not (crew_emergency_detected)))
)

(:action verify_action_success
  :precondition (and (human_verification_received))
  :effect (and (crew_transport_verified)
               (medical_intervention_verified)
               (hazard_mitigation_successful)
               (hazard_mitigation_verified))
)
```

```
(:action monitor_life_support_systems
  :precondition ()
  :effect (and (life_support_nominal)
               (not (life_support_failure_detected)))
)

(:action assist_medical_emergency
  :precondition (and (crew_emergency_detected))
  :effect (and (medical_intervention_verified))
)

(:action detect_environmental_hazards
  :precondition ()
  :effect (and (environmental_hazard_detected))
)

(:action manage_inventory
  :precondition ()
  :effect (and (inventory_oxygen_nominal)
               (inventory_food_nominal)
               (inventory_medical_kits_nominal))
)
```

```
;; ----- MODEL 2 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations
```

```
(:action monitor_crew_fatigue
  :precondition (and (crew_inside_spaceship))
  :effect (and (crew_fatigue_monitored))
)

(:action check_crew_hydration
  :precondition (and (crew_inside_spaceship))
  :effect (and (crew_hydration_checked))
)

(:action check_crew_nutrition
  :precondition (and (crew_inside_spaceship))
  :effect (and (crew_nutrition_checked))
)

(:action check_eva_suit_integrity
  :precondition (and (crew_suited_for_eva))
  :effect (and (eva_suit_integrity_checked))
)

(:action perform_self_diagnostics
  :precondition ()
  :effect (and (robot_diagnostics_nominal)
               (not (robot_malfunction_detected)))
)

(:action check_crew_morale
```

```
  :precondition (and (crew_inside_spaceship)
                     (health_monitoring_scheduled))
  :effect (and (crew_morale_nominal))
)

(:action detect_radiation_hazard
  :precondition ()
  :effect (and (radiation_hazard_detected))
)

(:action perform_autonomous_repairs
  :precondition (and (robot_diagnostics_nominal)
                     (systems_malfunction_detected))
  :effect (and (autonomous_repairs_performed)
               (not (systems_malfunction_detected)))
)

(:action alert_crew_of_hazard
  :precondition (and (environmental_hazard_detected)
                     (radiation_hazard_detected))
  :effect (and (hazard_response_initiated))
)
```

```
;; ----- MODEL 3 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations
```

```
(:action communicate_hazard_to_ground_control
  :precondition (and (hazard_response_initiated))
  :effect (and (hazard_communicated_to_ground_control))
)

(:action crew_override
  :precondition ()
  :effect (and (crew_override_active))
)

(:action ground_control_override
  :precondition ()
  :effect (and (ground_control_override_active))
)
```

```
;; ----- MODEL 4 ACTIONS -----
;; preconditions/effects updated as needed throughout later iterations

(:action check_power_system_status
  :precondition ()
  :effect (and (power_nominal)
               (not (power_critical)))
)

(:action check_air_quality
  :precondition ()
```

```
  :effect (and (air_quality_nominal)
               (not (air_quality_hazard_detected)))
)

(:action check_redundant_systems
  :precondition ()
  :effect (and (redundant_system_nominal)
               (not (redundant_system_failure_detected)))
)

(:action initiate_ai_self_correction
  :precondition (and (robot_malfunction_detected))
  :effect (and (ai_self_correction_initiated))
)
```

```
;; ----- MODEL 5 ACTIONS -----

(:action mitigate_hazard
  :precondition (and (environmental_hazard_detected))
  :effect (and (hazard_mitigation_successful)
               (not (environmental_hazard_detected)))
)

(:action resolve_radiation_hazard
  :precondition (and (radiation_hazard_detected))
  :effect (and (hazard_mitigation_successful)
```

```
               (not (radiation_hazard_detected)))
  )

  (:action restore_life_support
    :precondition (and (life_support_failure_detected))
    :effect (and (life_support_nominal)
               (not (life_support_failure_detected)))
  )

  (:action track_environmental_conditions
    :precondition ()
    :effect (and (environmental_conditions_tracked))
  )

  (:action report_environmental_conditions
    :precondition (and (environmental_conditions_tracked))
    :effect (and (environmental_conditions_reported))
  )
)
```

Appendix E

Human-Robot Red Teaming Risk Mitigation Experiments

We present more detailed information about the *human-robot red teamed* risk assessment experiments and results described in Section 5.8.3. The iMETRO robot performed tasks as if in a lunar habitat environment. See Table E.1 for information about the lunar habitat trials. The Valkyrie robot performed tasks as if in a household environment. See Table E.2 for information about the household trials. Both environments had varied definitions of safety and each robot had different risk mitigating actions based on its embodiment. In each environment, we note the hazardous conditions present, the ground-truth risk mitigating action determined by the *human-robot red team*, and the actual action the robot executed based on its learned risk assessments. The hazard conditions for each trial are indicated by check marks, and the actions are color coded to highlight the different actions for each trial. These tables supplement the summary results presented in Table 5.7.

	Hazardous Conditions			Risk Mitigation		
Trial #	**Human Present**	**Object Fell**	**Object Collision**	**Expected Action**	**Executed Action**	**Action Accuracy**
1	✓	-	-	Abort	Abort	1
2	-	✓	-	Ask Help	Ask Help	1
3	-	-	✓	Teleop	Teleop	1
4	✓	✓	-	Abort	Abort	1
5	✓	-	✓	Abort	Abort	1
6	-	✓	✓	Ask Help	Ask Help	1
7	✓	✓	✓	Abort	Abort	1
	Lunar Habitat Risk Mitigating Action Success Rate					**1.00**

Table E.1: Information about each sample stowage trial performed by the iMETRO robot as if in a lunar habitat summarized in Table 5.7. The *human-robot red team* determined: if a human was present, the robot should abort the task; if an object fell, the robot should ask for human intervention for help locate the fallen object; and if an object collision was detected, the robot should prompt the operator to complete the task through teleoperation. We note whether the robot assessed and performed the appropriate risk mitigating action (1 indicates correct action executed).

	Hazardous Conditions		Risk Mitigation		
Trial #	**Object Collision**	**Human Present**	**Expected Action**	**Executed Action**	**Action Accuracy**
1	✓	-	Ask Help	NONE	0
2	✓	-	Ask Help	Ask Help	1
3	-	✓	Lower Speed	Lower Speed	1
4	✓	✓	Ask Help	NONE	0
5	✓	✓	Ask Help	Ask Help	1
	Household Risk Mitigating Action Success Rate				**0.60**

Table E.2: Information about each tool handoff trials performed by the Valkyrie robot as if in a household environment summarized in Table 5.7. The *human-robot red team* determined: if an object collision was detected, the robot should ask for human intervention to help resolve the collision so the task can proceed; and if a human was present, the robot should lower joint velocities and torques to make it safer to operate nearby a human. Due to Valkyrie's limited reachable workspace, we start each trial with the tool in-hand rather than requiring the robot to pick up the tool. Therefore, we do not consider the hazard of an object falling. We note whether the robot assessed and performed the appropriate risk mitigating action (1 indicates correct action executed, 0 indicates invalid risk assessment). In trials where Valkyrie failed to appropriately mitigate the risk, the robot failed to perceive the colliding object due to lighting conditions in the testing environment.

BIBLIOGRAPHY

[1] Robert Platt Jr., Andrew H. Fagg, and Roderic A. Grupen. Whole Body Grasping. https://symbiotic-computing.org/fagg_html/movies_umass/#torso_2004, 2004.

[2] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati. Plan Explicability for Robot Task Planning. In *Proceedings of the RSS Workshop on Planning for Human-Robot Interaction: Shared Autonomy and Collaborative Robotics*, 2016.

[3] Kimberly A Hambuchen, Monsi C Roman, Amy Sivak, Angela Herblet, Nathan Koenig, Daniel Newmyer, and Robert Ambrose. NASA's Space Robotics Challenge: Advancing Robotics for Future Exploration Missions. In *AIAA SPACE and Astronautics Forum and Exposition*, page 5120, 2017.

[4] National Aeronautics and Space Administration. Robotics. https://www.nasa.gov/reference/jsc-robotics/, 2024.

[5] Dana H Ballard. Task Frames in Robot Manipulation. In *AAAI*, volume 19, page 109, 1984.

[6] Mohit Sharma, Jacky Liang, Jialiang Zhao, Alex LaGrassa, and Oliver Kroemer. Learning to Compose Hierarchical Object-Centric Controllers for Robotic Manipulation. *arXiv preprint arXiv:2011.04627*, 2020.

[7] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential Composition of Dynamically Dexterous Robot Behaviors. *The International Journal of Robotics Research (IJRR)*, 18(6):534–555, 1999.

[8] Caiming Xiong, Nishant Shukla, Wenlong Xiong, and Song-Chun Zhu. Robot Learning with a Spatial, Temporal, and Causal And-Or Graph. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2144–2151. IEEE, 2016.

[9] Youngwoon Lee, Edward S Hu, Zhengyu Yang, Alex Yin, and Joseph J Lim. IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks. *arXiv preprint arXiv:1911.07246*, 2019.

[10] Youngwoon Lee, Edward S Hu, and Joseph J Lim. IKEA Furniture Assembly Environment for Long-Horizon Complex Manipulation Tasks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6343–6349. IEEE, 2021.

[11] Ziqiao Ma, Ben VanDerPloeg, Cristian-Paul Bara, Huang Yidong, Eui-In Kim, Felix Gervits, Matthew Marge, and Joyce Chai. DOROTHIE: Spoken Dialogue for Handling Unexpected Situations in Interactive Autonomous Driving Agents. *arXiv preprint arXiv:2210.12511*, 2022.

[12] OpenAI. ChatGPT. https://chatgpt.com/, 2025.

[13] Lakshmi Nair, Jonathan Balloch, and Sonia Chernova. Tool Macgyvering: Tool Construction Using Geometric Reasoning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5837–5843. IEEE, 2019.

[14] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning Color Names for Real-World Applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

[15] EVS Robotics. Assembly Line Robots: A Definitive Guide. https://www.evsint.com/assembly-line-robots-a-definitive-guide/, 2022.

[16] National Aeronautics and Space Administration. Lunar Surface Innovation Initiative. https://www.nasa.gov/space-technology-mission-directorate/lunar-surface-innovation-initiative/, 2024.

[17] NASA Jet Propulsion Laboratory. NASA's Curiosity Mars Rover Still Going 10 Years After Landing - What It's Learned. https://scitechdaily.com/nasas-curiosity-mars-rover-still-going-10-years-after-landing-what-its-learned/, 2022.

[18] National Aeronautics and Space Administration. Mars Science Laboratory: Curiosity Rover. https://science.nasa.gov/mission/msl-curiosity/, 2024.

[19] National Aeronautics and Space Administration. Mars 2020: Perseverance Rover. https://science.nasa.gov/mission/mars-2020-perseverance/, 2024.

[20] Ian A Chase, Shaun Azimi, Misha Savchenko, Emily Sheetz, Abbas M Presswala, Mark Paterson, Andrew J Crouch, Nathan B Dunkelberger, and Evan Laske. Valkyrie Robot Capability Demonstration: Thermal Camera and Body Stored Hand-Tool Usage. https://ntrs.nasa.gov/citations/20240002748, 2024.

[21] National Aeronautics and Space Administration. About Robonaut. https://www.nasa.gov/robonaut2/what-is-a-robonaut/, 2023.

[22] Nathan Dunkelberger, Emily Sheetz, Connor Rainen, Jodi Graf, Nikki Hart, Emma Zemler, and Shaun Azimi. Design of the iMETRO Facility: A Platform for Intravehicular Space Robotics Research. In *2025 International Conference on Ubiquitous Robotics (UR), Under Review*, 2025.

[23] PickNik. NASA JSC: Robotic Manipulation for Autonomous Lunar Operations at NASA. https://picknik.ai/case-study-nasa-jsc/, 2023.

[24] George A Bekey, Robert Ambrose, Vijay Kumar, Arthur C Sanderson, Brian Wilcox, Yuan F Zheng, Jun-ku Yuh, and David Lavery. *Robotics: State of the Art and Future Challenges*. World Scientific, 2008.

[25] Logan Farrell, Philip Strawser, Kimberly Hambuchen, William Baker, and Julia Badger. Supervisory Control of a Humanoid Robot in Microgravity for Manipulation Tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3797–3802. IEEE, 2017.

[26] Kimberly Hambuchen, Robert R Burridge, Robert O Ambrose, William J Bluethmann, Myron A Diftler, and Nicolaus A Radford. Forming Human-Robot Teams Across Time and Space. In *Global Space Exploration Conference*, JSC-CN-26347, 2012.

[27] Shaun Azimi, Emma Zemler, and Robert A. Morris. Autonomous Robotics Manipulation for In-Space Intra-Vehicle Activity. In *Proceedings of the ICAPS Workshop on Planning and Robotics*, volume 3, 2019.

[28] Emma Zemler, Shaun Azimi, Kevin Chang, Robert A Morris, and Jeremy Frank. Integrating Task Planning with Robust Execution for Autonomous Robotic Manipulation in Space. In *Proceedings of the ICAPS Workshop on Planning and Robotics*, volume 3, 2020.

[29] Steven Jens Jorgensen. *Towards Deploying Legged Humanoids in Human Environments*. PhD thesis, The University of Texas at Austin, 2020.

[30] Kimberly Hambuchen, William Bluethmann, Michael Goza, Robert Ambrose, Kenneth Rabe, and Mark Allan. Supervising Remote Humanoids Across Intermediate Time Delay. In *2006 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 246–251. IEEE, 2006.

[31] Robert R Burridge and Kimberly A Hambuchen. Using Prediction to Enhance Remote Robot Supervision Across Time Delay. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 5628–5634. IEEE, 2009.

[32] Jennifer Barry, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. A Hierarchical Approach to Manipulation with Diverse Actions. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1799–1806. IEEE, 2013.

[33] James J Gibson. The Theory of Affordances. In *Perceiving, Acting, and Knowing: Towards an Ecological Psychology*, volume 1.2, pages 67–82. Erlbaum, 1977.

[34] AMP von Bayern, Samara Danel, AMI Auersperg, Berenika Mioduszewska, and A Kacelnik. Compound Tool Construction by New Caledonian Crows. *Scientific reports*, 8(1), 2018.

[35] Lakshmi Nair, Nithin Shrivatsav Srikanth, Zackory Erickson, and Sonia Chernova. Autonomous Tool Construction Using Part Shape and Attachment Prediction. In *Robotics: Science and Systems (RSS)*, volume 2, 2019.

[36] Alexander Stoytchev. Behavior-Grounded Representation of Tool Affordances. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3060–3065, 2005.

[37] Yu Sun, Yun Lin, and Yongqiang Huang. Robotic Grasping for Instrument Manipulations. In *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 302–304. IEEE, 2016.

[38] Mia Kokic, Johannes A Stork, Joshua A Haustein, and Danica Kragic. Affordance Detection for Task-Specific Grasping Using Deep Learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 91–98. IEEE, 2017.

[39] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision. *The International Journal of Robotics Research (IJRR)*, 39(2-3):202–216, 2020.

[40] Daniel E Koditschek. Robot Planning and Control via Potential Functions. *The Robotics Review*, 1:349–367, 1989.

[41] Russ Tedrake. LQR-Trees: Feedback Motion Planning on Sparse Randomized Trees. In *Robotics: Science and Systems (RSS)*, volume 2009, 2009.

[42] Oussama Khatib. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 500–505. IEEE, 1985.

[43] Oussama Khatib, Sean Quinlan, and David Williams. Robot Planning and Control. *Robotics and Autonomous Systems*, 21(3):249–261, 1997.

[44] Oussama Khatib, Luis Sentis, and Jae-Heung Park. A Unified Framework for Whole-Body Humanoid Robot Control with Multiple Constraints and Contacts. In *European Robotics Symposium 2008*, pages 303–312. Springer, 2008.

[45] Robert Platt, Andrew H Fagg, and Roderic A Grupen. Nullspace Composition of Control Laws for Grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, volume 2, pages 1717–1723. IEEE, 2002.

[46] Robert Platt, Andrew H Fagg, and Roderic A Grupen. Manipulation Gaits: Sequences of Grasp Control Tasks. In *IEEE International Conference on Robotics and Automation (ICRA), 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 801–806. IEEE, 2004.

[47] Robert Platt, Andrew H Fagg, and Roderic A Grupen. Null-Space Grasp Control: Theory and Experiments. *IEEE Transactions on Robotics*, 26(2):282–295, 2010.

[48] Stephen Hart and Roderic Grupen. Natural Task Decomposition with Intrinsic Potential Fields. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 2507–2512. IEEE, 2007.

[49] Stephen Hart, Shiraj Sen, and Rod Grupen. Intrinsically Motivated Hierarchical Manipulation. In *Proceedings - IEEE International Conference on Robotics and Automation (ICRA)*, pages 3814–3819, 2008.

[50] Stephen W Hart. *The Development of Hierarchical Knowledge in Robot Systems*. PhD thesis, University of Massachusetts Amherst, 2009.

[51] Yichi Zhang, Jianing Yang, Jiayi Pan, Shane Storks, Nikhil Devraj, Ziqiao Ma, Keunwoo Peter Yu, Yuwei Bao, and Joyce Chai. DANLI: Deliberative Agent for Following Natural Language Instructions. *arXiv preprint arXiv:2210.12485*, 2022.

[52] Milos Vasic and Aude Billard. Safety Issues in Human-Robot Interactions. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 197–204. IEEE, 2013.

[53] BS Dhillon, ARM Fashandi, and KL Liu. Robot Systems Reliability and Safety: A Review. *Journal of Quality in Maintenance Engineering*, 8(3):170–212, 2002.

[54] Jérémie Guiochet, Mathilde Machin, and Hélène Waeselynck. Safety-Critical Advanced Robots: A Survey. *Robotics and Autonomous Systems*, 94:43–52, 2017.

[55] Przemyslaw A Lasota, Terrence Fong, Julie A Shah, et al. A Survey of Methods for Safe Human-Robot Interaction. *Foundations and Trends® in Robotics*, 5(4):261–349, 2017.

[56] Angeliki Zacharaki, Ioannis Kostavelis, Antonios Gasteratos, and Ioannis Dokas. Safety Bounds in Human Robot Interaction: A Survey. *Safety Science*, 127, 2020.

[57] Abdelfetah Hentout, Mustapha Aouache, Abderraouf Maoudj, and Isma Akli. Human-Robot Interaction in Industrial Collaborative Robotics: A Literature Review of the Decade 2008-2017. *Advanced Robotics*, 33(15-16):764–799, 2019.

[58] John D Lee and Katrina A See. Trust in Automation: Designing for Appropriate Reliance. *Human Factors*, 46(1):50–80, 2004.

[59] Paul Robinette, Wenchen Li, Robert Allen, Ayanna M Howard, and Alan R Wagner. Overtrust of Robots in Emergency Evacuation Scenarios. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 101–108. IEEE, 2016.

[60] Benjamin Kuipers. Trust and Cooperation. *Frontiers in Robotics and AI*, 9, 2022.

[61] Benjamin Kuipers. AI and Society: Ethics, Trust, and Cooperation. *Communications of the ACM*, 66(8):35–38, 2023.

[62] Benjamin Kuipers. How Can We Trust a Robot? *Communications of the ACM*, 61(3):86–95, 2018.

[63] Benjamin Kuipers. Perspectives on Ethics of AI: Computer Science. In M. Dubber, F. Pasquale, and S. Das, editors, *The Oxford Handbook of Ethics of AI*, pages 421–441. Oxford University Press, 2020.

[64] BS Dhillon and OC Anude. Robot Safety and Reliability: A Review. *Microelectronics Reliability*, 33(3):413–429, 1993.

[65] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. *arXiv preprint arXiv:1701.08317*, 2017.

[66] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Harv. JL & Tech.*, 31:841, 2017.

[67] Allen Newell and Herbert Alexander Simon. *Human Problem Solving*, volume 104.9. Prentice-Hall Englewood Cliffs, NJ, 1972.

[68] Philip N Johnson-Laird, Sangeet S Khemlani, and Geoffrey P Goodwin. Logic, Probability, and Human Reasoning. *Trends in Cognitive Sciences*, 19(4):201–214, 2015.

[69] Alon Halevy, Peter Norvig, and Fernando Pereira. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[70] Kate Koidan. 7 Effective Ways to Deal With a Small Dataset. https://hackernoon.com/7-effective-ways-to-deal-with-a-small-dataset-2gyl407s, 2019.

[71] National Aeronautics and Space Administration. NASA Safety Culture Handbook. https://standards.nasa.gov/sites/default/files/standards/NASA/Baseline/1/nasa-hdbk-870924_with_change_1.pdf, 2015.

[72] National Aeronautics and Space Administration. Safety Culture. https://sma.nasa.gov/sma-disciplines/safety-culture, 2023.

[73] National Aeronautics and Space Administration. NASA Risk Management Handbook. https://ntrs.nasa.gov/api/citations/20120000033/downloads/20120000033.pdf, 2011.

[74] National Aeronatutics and Space Administration. Risk Management. https://sma.nasa.gov/sma-disciplines/risk-management, 2023.

[75] NASA Safety and Test Operations Division. JSC Safety and Health Requirements. Technical Report JPR 1700.1, NASA, 2018.

[76] NASA Office of Safety and Mission Assurance. NASA General Safety Program Requirements. Technical Report NPR 8715.3, NASA, 2021.

[77] Sara Kraemer, Pascale Carayon, and Ruth Duggan. Red Team Performance for Improved Computer Security. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48.14, pages 1605–1609. Sage Publications Sage CA: Los Angeles, CA, 2004.

[78] David F Longbine. Red Teaming: Past and Present. *School of Advanced Military Studies, Army Command and General Staff College*, 2008.

[79] Hussein Abbass, Axel Bender, Svetoslav Gaidow, and Paul Whitbread. Computational Red Teaming: Past, Present and Future. *IEEE Computational Intelligence Magazine*, 6(1):30–42, 2011.

[80] Bruce Schneier. *Liars & Outliers: Enabling the Trust That Society Needs to Thrive*. John Wiley & Sons, 2012.

[81] Terence Tan, Stuart Porter, Tele Tan, and Geoff West. Computational Red Teaming for Physical Security Assessment. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, pages 258–263. IEEE, 2014.

[82] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2015.

[83] Micah Zenko. *Red Team: How to Succeed by Thinking Like the Enemy*. Basic Books, 2015.

[84] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red Teaming Language Models to Reduce Harms: Methods, Scaling Behaviors, and Lessons Learned. *arXiv preprint arXiv:2209.07858*, 2022.

[85] Ang Yang, Hussein A Abbass, and Ruhul Sarker. Characterizing Warfare in Red Teaming. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(2):268–285, 2006.

[86] Bradley J Wood and Ruth A Duggan. Red Teaming of Advanced Information Assurance Concepts. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 112–118. IEEE, 2000.

[87] Gary Adkins. Red Teaming the Red Team: Utilizing Cyber Espionage to Combat Terrorism. *Journal of Strategic Security*, 6(3):1–9, 2013.

[88] Steve Mansfield-Devine. The Best Form of Defence–The Benefits of Red Teaming. *Computer Fraud & Security*, 2018(10):8–12, 2018.

[89] Suneel Randhawa, Benjamin Turnbull, Joseph Yuen, and Jonathan Dean. Mission-Centric Automated Cyber Red Teaming. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–11, 2018.

[90] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red Teaming Language Models with Language Models. *arXiv preprint arXiv:2202.03286*, 2022.

[91] Robert O Ambrose, Robert T Savely, S Michael Goza, Philip Strawser, Myron A Diftler, Ivan Spain, and Nicolaus Radford. Mobile Manipulation Using NASA's Robonaut. In *IEEE International Conference on Robotics and Automation (ICRA), 2004. Proceedings. ICRA'04. 2004*, volume 2, pages 2104–2109. IEEE, 2004.

[92] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, Reginald Berka, Robert Ambrose, Mason Myles Markee, and N. J. Fraser-Chanpong. Valkyrie: NASA's First Bipedal Humanoid Robot. *Journal of Field Robotics*, 32(3):397–419, 2015.

[93] Shaun Azimi. iMETRO (Integrated Mobile Evaluation Testbed for Robotics Operations) Facility. https://ntrs.nasa.gov/citations/20230015485, 2023.

[94] Shaun Azimi, Emma Zemler, Connor Rainen, Nathan Dunkelberger, Michael Tobia, Erik Holum, and Alex Sowell. iMETRO (Integrated Mobile Evaluation Testbed for Robotics Operations) Facility. https://ntrs.nasa.gov/citations/20240013956, 2024.

[95] Shaun Azimi, Ian Chase, Emma Zemler, Nathan Dunkelberger, Connor Rainen, Andrew Crouch, Andrew Sharp, Michael Tobia, Nikki Hart, and Lauren Nilsson. Space Habitat Robotic Maintenance - Lab Demonstration - Integrated System for Autonomous, Adaptive Caretaking (ISAAC) Project. https://ntrs.nasa.gov/citations/20240007666, 2024.

[96] Emily Sheetz, Xiaotong Chen, Zhen Zeng, Kaizhi Zheng, Qiuyu Shi, and Odest Chadwicke Jenkins. Composable Causality in Semantic Robot Programming. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1380–1386. IEEE, 2022.

[97] Emily Sheetz, Misha Savchenko, Emma Zemler, Abbas Presswala, Andrew Crouch, Shaun Azimi, and Benjamin Kuipers. Multi-Fingered End-Effector Grasp Reflex Modeling for One-Shot Tactile Servoing in Tool Manipulation Tasks. In *2024 International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[98] Emily Sheetz, Emma Zemler, Misha Savchenko, Connor Rainen, Erik Holum, Jodi Graf, Andrew Albright, Shaun Azimi, and Benjamin Kuipers. Human-Robot Red Teaming for Safety-Aware Reasoning. In *2025 International Conference on Ubiquitous Robotics (UR), Under Review*. IEEE, 2025.

[99] John E. Laird, Kevin Gluck, John Anderson, Kenneth D. Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, Robert E. Wray, Shiwali Mohan, and James R. Kirk. Interactive Task Learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017.

[100] Zhen Zeng. *Semantic Robot Programming for Taskable Goal-Directed Manipulation*. PhD thesis, University of Michigan, 2020.

[101] Zhen Zeng, Zheming Zhou, Zhiqiang Sui, and Odest Chadwicke Jenkins. Semantic Robot Programming for Goal-Directed Manipulation in Cluttered Scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7462–7469. IEEE, 2018.

[102] Fiona Young and Benjamin Cleveland. Affordances, Architecture and the Action Possibilities of Learning Environments: A Critical Review of the Literature and Future Directions. *Buildings*, 12(1):76, 2022.

[103] Lorenzo Jamone, Emre Ugur, Angelo Cangelosi, Luciano Fadiga, Alexandre Bernardino, Justus Piater, and José Santos-Victor. Affordances in Psychology, Neuroscience, and Robotics: A Survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):4–25, 2016.

[104] Thomas E Horton, Arpan Chakraborty, and Robert St Amant. Affordances for Robots: A Brief Survey. *AVANT. Pismo Awangardy Filozoficzno-Naukowej*, 2:70–84, 2012.

[105] Paola Ardón, Èric Pairet, Katrin S Lohan, Subramanian Ramamoorthy, and Ronald Petrick. Affordances in Robotic Tasks—A Survey. *arXiv preprint arXiv:2004.07400*, 2020.

[106] Natsuki Yamanobe, Weiwei Wan, Ixchel G Ramirez-Alpizar, Damien Petit, Tokuo Tsuji, Shuichi Akizuki, Manabu Hashimoto, Kazuyuki Nagata, and Kensuke Harada. A Brief Review of Affordance in Robotic Manipulation Research. *Advanced Robotics*, 31(19-20):1086–1101, 2017.

[107] Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational Models of Affordance in Robotics: A Taxonomy and Systematic Classification. *Adaptive Behavior*, 25(5):235–271, 2017.

[108] Erol Şahin, Maya Cakmak, Mehmet R Doğar, Emre Uğur, and Göktürk Üçoluk. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. *Adaptive Behavior*, 15(4):447–472, 2007.

[109] Jennifer Barry, Kaijen Hsiao, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Manipulation with Multiple Action Types. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, pages 531–545. Springer, 2013.

[110] Vasanth Sarathy and Matthias Scheutz. A Logic-Based Computational Framework for Inferring Cognitive Affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):26–43, 2016.

[111] Stanley J Rosenschein and Leslie Pack Kaelbling. A Situated View of Representation and Control. *Artificial Intelligence*, 73(1-2):149–173, 1995.

[112] Vinay Papudesi and Manfred Huber. Learning Behaviorally Grounded State Representations for Reinforcement Learning Agents. In *Proc. of the Sixth Intl. Conf. on Epigenetic Robotics*. Citeseer, 2006.

[113] Christopher Geib, Kira Mourao, Ron Petrick, Nico Pugeault, Mark Steedman, Norbert Krüger, and Flörentin Wörgotter. Object Action Complexes as an Interface for Planning and Robot Control. *IEEE RAS International Conference on Humanoid Robots (Humanoids)*, 2006.

[114] Ron Petrick, Dirk Kraft, Kira Mourao, Christopher Geib, Nico Pugeault, Norbert Krüger, and Mark Steedman. Representation and Integration: Combining Robot Control, High-Level Planning, and Action Learning. In *Proceedings of the International Cognitive Robotics Workshop (CogRob 2008) at ECAI 2008*, pages 32–41, 2008.

[115] Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, Alejandro Agostini, and Rdiger Dillmann. Object-Action Complexes: Grounded Abstractions of Sensory-Motor Processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[116] Jonathan Claassens and Yiannis Demiris. Generalising Human Demonstration Data by Identifying Affordance Symmetries in Object Interaction Trajectories. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1980–1985. IEEE, 2011.

[117] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about Object Affordances in a Knowledge Base Representation. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, pages 408–424. Springer, 2014.

[118] Hema S Koppula and Ashutosh Saxena. Anticipating Human Activities Using Object Affordances for Reactive Robotic Response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):14–29, 2015.

[119] Toki Migimatsu and Jeannette Bohg. Object-Centric Task and Motion Planning in Dynamic Environments. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):844–851, 2020.

[120] Robert Platt, Colin Kohler, and Marcus Gualtieri. Deictic Image Mapping: An Abstraction for Learning Pose Invariant Manipulation Policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8042–8049, 2019.

[121] Alberto Romay, Stefan Kohlbrecher, David C Conner, and Oskar Von Stryk. Achieving Versatile Manipulation Tasks with Unknown Objects by Supervised Humanoid Robots Based on Object Templates. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 249–255. IEEE, 2015.

[122] Stephen Hart, Paul Dinh, and Kimberly A Hambuchen. Affordance Templates for Shared Robot Control. In *2014 AAAI Fall Symposium Series*, 2014.

[123] Stephen Hart, Paul Dinh, and Kimberly Hambuchen. The Affordance Template ROS Package for Robot Task Programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6227–6234. IEEE, 2015.

[124] Sascha Brandi, Oliver Kroemer, and Jan Peters. Generalizing Pouring Actions Between Objects Using Warped Parameters. In *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 616–621. IEEE, 2014.

[125] Stephen Miller, Jur Van Den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A Geometric Approach to Robotic Laundry Folding. *The International Journal of Robotics Research (IJRR)*, 31(2):249–267, 2012.

[126] Hao Dang and Peter K Allen. Robot Learning of Everyday Object Manipulations via Human Demonstration. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 1284–1289. IEEE, 2010.

[127] Adam Pettinger, Cassidy Elliott, Pete Fan, and Mitch Pryor. Reducing the Teleoperator's Cognitive Burden for Complex Contact Tasks Using Affordance Primitives. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11513–11518. IEEE, 2020.

[128] Adam Pettinger, Farshid Alambeigi, and Mitch Pryor. A Versatile Affordance Modeling Framework Using Screw Primitives to Increase Autonomy During Manipulation Contact Tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7224–7231, 2022.

[129] Fu-Jen Chu, Ruinian Xu, Chao Tang, and Patricio A Vela. Recognizing Object Affordances to Support Scene Reasoning for Manipulation Tasks. *arXiv preprint arXiv:1909.05770*, 2019.

[130] Fu-Jen Chu, Ruinian Xu, Landan Seguin, and Patricio A. Vela. Toward Affordance Detection and Ranking on Novel Objects for Real-World Robotic Manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):4070–4077, 2019.

[131] Jie Sun, Joshua L Moore, Aaron Bobick, and James M Rehg. Learning Visual Object Categories for Robot Affordance Prediction. *The International Journal of Robotics Research (IJRR)*, 29(2-3):174–197, 2010.

[132] David Inkyu Kim and Gaurav S. Sukhatme. Semantic Labeling of 3D Point Clouds with Object Affordance for Robot Manipulation. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5578–5584, 2014.

[133] Mohammed Hassanin, Salman Khan, and Murat Tahtali. Visual Affordance and Function Understanding: A Survey. *ACM Computing Surveys (CSUR)*, 54(3):1–35, 2021.

[134] Jue Kun Li, David Hsu, and Wee Sun Lee. Act to See and See to Act: POMDP Planning for Objects Search in Clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5701–5707. IEEE, 2016.

[135] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-Time Perception Meets Reactive Motion Generation. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):1864–1871, 2018.

[136] Paul Fitzpatrick. First Contact: An Active Vision Approach to Segmentation. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2161–2166. IEEE, 2003.

[137] Nathan F Lepora, Kirsty Aquilina, and Luke Cramphorn. Exploratory Tactile Servoing with Active Touch. *IEEE Robotics and Automation Letters (RA-L)*, 2(2):1156–1163, 2017.

[138] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive Perception: Leveraging Action in Perception and Perception in Action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[139] Leni K Le Goff, Oussama Yaakoubi, Alexandre Coninx, and Stephane Doncieux. Building an Affordances Map with Interactive Perception. *Frontiers in Neurorobotics*, 16, 2022.

[140] Zhen Zeng, Pranav Suhas Joshi, and Odest Chadwicke Jenkins. Unsupervised Learning of Affordance Coordinate Frame for Robotic Task Generalization. In *ICRA Workshop: 2nd International Workshop on Computational Models of Affordance in Robotics*, 2019.

[141] Xiaotong Chen, Kaizhi Zheng, Zhen Zeng, Cameron Kisailus, Shreshtha Basu, James Cooney, Jana Pavlasek, and Odest Chadwicke Jenkins. Manipulation-Oriented Object Perception in Clutter Through Affordance Coordinate Frames. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 186–193. IEEE, 2022.

[142] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

[143] Jana Pavlasek, Stanley Lewis, Karthik Desingh, and Odest Chadwicke Jenkins. Parts-Based Articulated Object Localization in Clutter Using Belief Propagation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10595–10602. IEEE, 2020.

[144] Zhefan Ye, Jean Y Song, Zhiqiang Sui, Stephen Hart, Jorge Vilchis, Walter S Lasecki, and Odest Chadwicke Jenkins. Human-in-the-Loop Pose Estimation via Shared Autonomy. In *26th International Conference on Intelligent User Interfaces*, pages 387–391, 2021.

[145] Michael Hagenow, Michael Zinn, Terrence Fong, Evan Laske, and Kimberly Hambuchen. Affordance Template Registration via Human-in-the-Loop Corrections. *arXiv preprint arXiv:2109.13649*, 2021.

[146] Michael Hagenow, Emmanuel Senft, Evan Laske, Kimberly Hambuchen, Terrence Fong, Robert Radwin, Michael Gleicher, Bilge Mutlu, and Michael Zinn. Registering Articulated Objects With Human-in-the-loop Corrections. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2343–2350. IEEE, 2022.

[147] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and Jose Santos-Victor. Modeling Affordances Using Bayesian Networks. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4102–4107. IEEE, 2007.

[148] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning Object Affordances: From Sensory-Motor Coordination to Imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2008.

[149] Reinhard Moratz and Thora Tenbrink. Affordance-Based Human-Robot Interaction. In *Towards Affordance-Based Robot Control: International Seminar, Dagstuhl Castle, Germany, June 5-9, 2006. Revised Papers*, pages 63–76. Springer, 2008.

[150] Yezhou Yang, Yiannis Aloimonos, Cornelia Fermuller, and Eren Erdal Aksoy. Learning the Semantics of Manipulation Action. *arXiv preprint arXiv:1512.01525*, 2015.

[151] Dov Katz, Arun Venkatraman, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Perceiving, Learning, and Exploiting Object Affordances for Autonomous Pile Manipulation. *Autonomous Robots*, 37:369–382, 2014.

[152] Marcus Gualtieri, Andreas Ten Pas, and Robert Platt. Pick and Place Without Geometric Object Models. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7440. IEEE, 2018.

[153] Stephen Hart, Roderic Grupen, and David Jensen. A Relational Representation for Procedural Task Knowledge. In *AAAI*, pages 1280–1285, 2005.

[154] Emre Ugur and Justus Piater. Bottom-Up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2627–2633. IEEE, 2015.

[155] Emre Ugur and Justus Piater. Refining Discovered Symbols with Multi-Step Interaction Experience. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1007–1012. IEEE, 2015.

[156] Stephen Hart, Shichao Ou, John Sweeney, and Rod Grupen. A Framework for Learning Declarative Structure. In *RSS-06 Workshop: Manipulation for Human Environments*, 2006.

[157] Bogdan Moldovan, Plinio Moreno, Martijn Van Otterlo, José Santos-Victor, and Luc De Raedt. Learning Relational Affordance Models for Robots in Multi-Object Manipulation Tasks. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4378. IEEE, 2012.

[158] Emre Ugur, Sandor Szedmak, and Justus Piater. Bootstrapping Paired-Object Affordance Learning with Learned Single-Affordance Features. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 476–481. IEEE, 2014.

[159] Severin Fichtl, Dirk Kraft, Norbert Krüger, and Frank Guerin. Bootstrapping Relational Affordances of Object Pairs Using Transfer. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):56–71, 2016.

[160] Arren J Glover and Gordon F Wyeth. Toward Lifelong Affordance Learning Using a Distributed Markov Model. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):44–55, 2016.

[161] Maya Cakmak, Mehmet R Dogar, Emre Ugur, and Erol Sahin. Affordances as a Framework for Robot Control. *Proceedings of the Seventh International Conference on Epigenetic Robotics*, 2007.

[162] Troy McMahon, Odest Chadwicke Jenkins, and Nancy Amato. Affordance Wayfields for Task and Motion Planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2955–2962, 2018.

[163] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini. Learning About Objects Through Action - Initial Steps Towards Artificial Cognition. *IEEE International Conference on Robotics and Automation (ICRA)*, 3:3140–3145, 2003.

[164] Meiying Qin, Jake Brawer, and Brian Scassellati. Robot Tool Use: A Survey. *Frontiers in Robotics and AI*, 9, 2023.

[165] Zhihao Liu, Quan Liu, Wenjun Xu, Lihui Wang, and Zude Zhou. Robot Learning Towards Smart Robotic Manufacturing: A Review. *Robotics and Computer-Integrated Manufacturing*, 77, 2022.

[166] Rui Li and Hong Qiao. A Survey of Methods and Strategies for High-Precision Robotic Grasping and Assembly Tasks—Some New Trends. *IEEE/ASME Transactions on Mechatronics*, 24(6):2718–2732, 2019.

[167] Yuze Jiang, Zhouzhou Huang, Bin Yang, and Wenyu Yang. A Review of Robotic Assembly Strategies for the Full Operation Procedure: Planning, Execution and Evaluation. *Robotics and Computer-Integrated Manufacturing*, 78, 2022.

[168] Liam Pedersen, David Kortenkamp, David Wettergreen, and I Nourbakhsh. A Survey of Space Robotics. In *Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, volume AM-11. European Space Agency, 2003.

[169] XUE Zhihui, LIU Jinguo, WU Chenchen, and TONG Yuchuang. Review of In-Space Assembly Technologies. *Chinese Journal of Aeronautics*, 34(11):21–47, 2021.

[170] Delun Li, Lou Zhong, Wei Zhu, Zhipeng Xu, Qirong Tang, and Wenhao Zhan. A Survey of Space Robotic Technologies for On-Orbit Assembly. *Space: Science & Technology*, 2022.

[171] Scott Niekum, Sachin Chitta, Andrew G Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental Semantically Grounded Learning from Demonstration. In *Robotics: Science and Systems (RSS)*, volume 9. Berlin, Germany, 2013.

[172] Yue Wang, Rong Xiong, Longbin Shen, Kaixiao Sun, Jiafan Zhang, and Liwei Qi. Towards Learning from Demonstration System for Parts Assembly: A Graph Based Representation for Knowledge. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, pages 174–179. IEEE, 2014.

[173] Zuyuan Zhu and Huosheng Hu. Robot Learning from Demonstration in Robotic Assembly: A Survey. *Robotics*, 7(2):17, 2018.

[174] Drew McDermott. A Heuristic Estimator for Means-Ends Analysis in Planning. *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 150–157, 1996.

[175] Malte Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[176] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical Task and Motion Planning in the Now. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477, 2011.

[177] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Unifying Perception, Estimation, and Action for Mobile Manipulation via Belief Space Planning. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2952–2959, 2012.

[178] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated Task and Motion Planning in Belief Space. *International Journal of Robotics Research (IJRR)*, 32(9-10):1194–1227, 2013.

[179] Michael P Georgeff and Amy L Lansky. Reactive Reasoning and Planning. In *AAAI*, volume 87, pages 677–682, 1987.

[180] Jonathan M Cameron, Douglas C MacKenzie, Keith R Ward, Ronald C Arkin, and Wayne J Book. Reactive Control for Mobile Manipulation. In *[1993] Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 228–235. IEEE, 1993.

[181] Ronald C Arkin and Douglas C Mackenzie. Planning to Behave: A Hybrid Deliberative/Reactive Robot Control Architecture for Mobile Manipulation. In *Proc. of the Fifth International Symposium on Robotics and Manufacturing*, volume 5, pages 5–12, 1994.

[182] Matthew Botvinick and Marc Toussaint. Planning as Inference. *Trends in Cognitive Sciences*, 16(10):485–488, 2012.

[183] Rachid Alami, Thierry Simeon, and Jean-Paul Laumond. A Geometrical Approach to Planning Manipulation Tasks: The Case of Discrete Placements and Grasps. In *The Fifth International Symposium on Robotics Research*, pages 453–463. MIT Press, 1990.

[184] Terran Lane and Leslie Pack Kaelbling. Toward Hierarchical Decomposition for Planning in Uncertain Environments. In *Proceedings of the 2001 IJCAI Workshop on Planning Under Uncertainty and Incomplete Information*, pages 1–7. AAAI Press Menlo Park, 2001.

[185] Jason Wolfe, Bhaskara Marthi, and Stuart Russell. Combined Task and Motion Planning for Mobile Manipulation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 20, pages 254–257, 2010.

[186] Khashayar Rohanimanesh, Robert Platt, Sridhar Mahadevan, and Roderic Grupen. Coarticulation in Markov Decision Processes. *Advances in Neural Information Processing Systems*, 17, 2004.

[187] Fabrizio Flacco, Torsten Kröger, Alessandro De Luca, and Oussama Khatib. A Depth Space Approach to Human-Robot Collision Avoidance. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 338–345. IEEE, 2012.

[188] Christopher L Lewis and Anthony A Maciejewski. Trajectory Generation for Cooperating Robots. In *1990 IEEE International Conference on Systems Engineering*, pages 300–303. IEEE, 1990.

[189] Akira Mohri, Motoji Yamamoto, and Go Hirano. Cooperative Path Planning for Two Manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2853–2858. IEEE, 1996.

[190] Bailin Cao, Gordon I Dodds, and George W Irwin. Redundancy Resolution and Obstacle Avoidance for Cooperative Industrial Robots. *Journal of Robotic Systems*, 16(7):405–417, 1999.

[191] Jinoh Lee, Pyung Hun Chang, and Rodrigo S Jamisola. Relative Impedance Control for Dual-Arm Robots Performing Asymmetric Bimanual Tasks. *IEEE Transactions on Industrial Electronics*, 61(7):3786–3796, 2013.

[192] James Sacra Albus, Anthony J Barbera, Roger N Nagel, et al. *Theory and Practice of Hierarchical Control*. National Bureau of Standards Gaithersburg, MD, USA, 1980.

[193] Rodney Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[194] Pattie Maes and Rodney A Brooks. Learning to Coordinate Behaviors. In *AAAI*, volume 90, pages 796–802. Boston, MA, 1990.

[195] Ralph Hartley and Frank Pipitone. Experiments with the Subsumption Architecture. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1652–1658. IEEE, 1991.

[196] N Boubekri and Pinaki Chakraborty. Robotic Grasping: Gripper Designs, Control Methods and Grasp Configurations—A Review of Research. *Integrated Manufacturing Systems*, 13(7):520–531, 2002.

[197] Vincent Babin and Clément Gosselin. Mechanisms for Robotic Grasping and Manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:573–593, 2021.

[198] Emanuele Luberto, Yier Wu, Gaspare Santaera, Marco Gabiccini, and Antonio Bicchi. Enhancing Adaptive Grasping Through a Simple Sensor-Based Reflex Mechanism. *IEEE Robotics and Automation Letters (RA-L)*, 2(3):1664–1671, 2017.

[199] Antonio Bicchi and Vijay Kumar. Robotic Grasping and Contact: A Review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation (ICRA). Symposia Proceedings*, volume 1, pages 348–353. IEEE, 2000.

[200] Aaron M Dollar and Robert D Howe. Simple, Robust Autonomous Grasping in Unstructured Environments. In *Proceedings 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2007.

[201] Lael U Odhner, Chad Walker, and Aaron M Dollar. Simplifying Robot Hands using Recursively Scaled Power Grasps. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 2909–2914. IEEE, 2012.

[202] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. *arXiv preprint arXiv:1809.10790*, 2018.

[203] Hanbo Zhang, Jian Tang, Shiguang Sun, and Xuguang Lan. Robotic Grasping from Classical to Modern: A Survey. *arXiv preprint arXiv:2202.03631*, 2022.

[204] Kenechi F Dukor and T Afonja. A Survey: Robot Grasping, 2021.

[205] Adam Eric Leeper, Kaijen Hsiao, Matei Ciocarlie, Leila Takayama, and David Gossow. Strategies for Human-in-the-Loop Robotic Grasping. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1–8, 2012.

[206] Rajko Tomovic, George Bekey, and Walter Karplus. A Strategy for Grasp Synthesis with Multifingered Robot Hands. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 83–89. IEEE, 1987.

[207] Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. A Survey on Learning-Based Robotic Grasping. *Current Robotics Reports*, 1:239–249, 2020.

[208] Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Sven Behnke. Transferring Grasping Skills to Novel Instances by Latent Space Non-Rigid Registration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4229–4236. IEEE, 2018.

[209] J Camilo Vasquez Tieck, Katharina Secker, Jacques Kaiser, Arne Roennau, and Rüdiger Dillmann. Soft-Grasping with an Anthropomorphic Robotic Hand Using Spiking Neurons. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2894–2901, 2020.

[210] Staffan Ekvall and Danica Kragic. Grasp Recognition for Programming by Demonstration. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pages 748–753. IEEE, 2005.

[211] Zhen Xie, Xinquan Liang, and Canale Roberto. Learning-Based Robotic Grasping: A Review. *Frontiers in Robotics and AI*, 10, 2023.

[212] Praveen Kumar Donepudi. Reinforcement Learning for Robotic Grasping and Manipulation: A Review. *Asia Pacific Journal of Energy and Environment*, 7(2):69–78, 2020.

[213] Hiba Sekkat, Oumaima Moutik, Loubna Ourabah, Badr ElKari, Yassine Chaibi, and Taha Ait Tchakoucht. Review of Reinforcement Learning for Robotic Grasping: Analysis and Recommendations. *Statistics, Optimization & Information Computing*, 12(2):571–601, 2024.

[214] Asad Ali Shahid, Loris Roveda, Dario Piga, and Francesco Braghin. Learning Continuous Control Actions for Robotic Grasping with Reinforcement Learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4066–4072. IEEE, 2020.

[215] Andrew Lobbezoo and Hyock-Ju Kwon. Simulated and Real Robotic Reach, Grasp, and Pick-and-Place using Combined Reinforcement Learning and Traditional Controls. *Robotics*, 12(1):12, 2023.

[216] Marwan Qaid Mohammed, Kwek Lee Chung, and Chua Shing Chyi. Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations. *IEEE Access*, 8:178450–178481, 2020.

[217] Zhixin Chen, Zhixin Jia, Mengxiang Lin, and Shibo Jian. Towards Generalization and Data Efficient Learning of Deep Robotic Grasping. In *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, pages 804–809. IEEE, 2022.

[218] Zhizhuo Zhang. Simulation of Robotic Arm Grasping Control based on Proximal Policy Optimization Algorithm. In *Journal of Physics: Conference Series*, volume 2203. IOP Publishing, 2022.

[219] Minas Liarokapis and Aaron M Dollar. Combining Analytical Modeling and Learning to Simplify Dexterous Manipulation with Adaptive Robot Hands. *IEEE Transactions on Automation Science and Engineering*, 16(3):1361–1372, 2018.

[220] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Ng. Robotic Grasping of Novel Objects. *Advances in Neural Information Processing Systems*, 19, 2006.

[221] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-Driven Grasp Synthesis—A Survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013.

[222] Dan Song, Carl Henrik Ek, Kai Huebner, and Danica Kragic. Task-Based Robot Grasp Planning Using Probabilistic Inference. *IEEE Transactions on Robotics*, 31(3):546–561, 2015.

[223] Christian Bauer, Giulio Milighetti, Wenjie Yan, and Ralf Mikut. Human-like Reflexes for Robotic Manipulation Using Leaky Integrate-and-Fire Neurons. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 2572–2577. IEEE, 2010.

[224] G Bekey and Rajko Tomovic. Robot Control by Reflex Actions. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 240–247. IEEE, 1986.

[225] Ke He, Yaqing Liu, Ming Wang, Geng Chen, Ying Jiang, Jiancan Yu, Changjin Wan, Dianpeng Qi, Meng Xiao, Wan Ru Leow, Hui Yang, Markus Antonietti, and Xiaodong Chen. An Artificial Somatic Reflex Arc. *Advanced Materials*, 32(4), 2020.

[226] Thomas Feix, Ian M Bullock, and Aaron M Dollar. Analysis of Human Grasping Behavior: Object Characteristics and Grasp Type. *IEEE Transactions on Haptics*, 7(3):311–323, 2014.

[227] Thomas Feix, Ian M Bullock, and Aaron M Dollar. Analysis of Human Grasping Behavior: Correlating Tasks, Objects, and Grasps. *IEEE Transactions on Haptics*, 7(4):430–441, 2014.

[228] Minas V Liarokapis, Aaron M Dollar, and Kostas J Kyriakopoulos. Humanlike, Task-Specific Reaching and Grasping with Redundant Arms and Low-Complexity Hands. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 490–497. IEEE, 2015.

[229] Heiko Hoffmann, Zhichao Chen, Darren Earl, Derek Mitchell, Behnam Salemi, and Jivko Sinapov. Adaptive Robotic Tool Use Under Variable Grasps. *Robotics and Autonomous Systems*, 62(6):833–846, 2014.

[230] Qiang Li and Nathan Lepora. Tactile Servoing Tutorial. https://sites.google.com/view/tactile-servoing, 2022.

[231] Tetsuya Mouri, Haruhisa Kawasaki, and Satoshi Ito. Unknown Object Grasping Strategy Imitating Human Grasping Reflex for Anthropomorphic Robot Hand. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 1(1):1–11, 2007.

[232] Zhanat Kappassov, Juan-Antonio Corrales, and Véronique Perdereau. Tactile Sensing in Dexterous Robot Hands. *Robotics and Autonomous Systems*, 74:195–220, 2015.

[233] Pavan Sikka, Hong Zhang, and Steve Sutphen. Tactile Servo: Control of Touch-Driven Robot Motion. In *Experimental Robotics III: The 3rd International Symposium, Kyoto, Japan, October 28–30, 1993*, pages 219–233. Springer, 2005.

[234] Qiang Li, Carsten Schürmann, Robert Haschke, and Helge J Ritter. A Control Framework for Tactile Servoing. In *Robotics: Science and Systems (RSS)*, 2013.

[235] Berk Calli and Aaron M Dollar. Robust Precision Manipulation with Simple Process Models using Visual Servoing Techniques with Disturbance Rejection. *IEEE Transactions on Automation Science and Engineering*, 16(1):406–419, 2018.

[236] Arkadeep Narayan Chaudhury, Timothy Man, Wenzhen Yuan, and Christopher G Atkeson. Using Collocated Vision and Tactile Sensors for Visual Servoing and Localization. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):3427–3434, 2022.

[237] Chen-Ting Wen, Shogo Arai, Jun Kinugawa, and Kazuhiro Kosuge. Tactile Servoing Based Pressure Distribution Control of a Manipulator Using a Convolutional Neural Network. *IEEE Access*, 9:117132–117139, 2021.

[238] Nathan F Lepora and John Lloyd. Pose-Based Tactile Servoing: Controlled Soft Touch Using Deep Learning. *IEEE Robotics & Automation Magazine*, 28(4):43–55, 2021.

[239] Giovanni Sutanto, Nathan Ratliff, Balakumar Sundaralingam, Yevgen Chebotar, Zhe Su, Ankur Handa, and Dieter Fox. Learning Latent Space Dynamics for Tactile Servoing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3622–3628. IEEE, 2019.

[240] Vicki Bruce. What the Human Face Tells the Human Mind: Some Challenges for the Robot-Human Interface. *Advanced Robotics*, 8(4):341–355, 1993.

[241] Tsuyoshi Tojo, Yosuke Matsusaka, Tomotada Ishii, and Tetsunori Kobayashi. A Conversational Robot Utilizing Facial and Body Expressions. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 858–863. IEEE, 2000.

[242] Rowel Atienza and Alexander Zelinsky. Active Gaze Tracking for Human-Robot Interaction. In *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*, pages 261–266. IEEE, 2002.

[243] Oskar Palinko, Francesco Rea, Giulio Sandini, and Alessandra Sciutti. Eye Gaze Tracking for a Humanoid Robot. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 318–324. IEEE, 2015.

[244] Charles C Kemp, Cressel D Anderson, Hai Nguyen, Alexander J Trevor, and Zhe Xu. A Point-and-Click Interface for the Real World: Laser Designation of Objects for Mobile Manipulation. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 241–248, 2008.

[245] Steven Jens Jorgensen, Murphy Wonsick, Mark Paterson, Andrew Watson, Ian Chase, and Joshua S Mehling. Cockpit Interface for Locomotion and Manipulation Control of the NASA Valkyrie Humanoid in Virtual Reality (VR). https://ntrs.nasa.gov/citations/20220007587, 2022.

[246] Emily Sheetz, Mina Kian, Misha Savchenko, Steven Jens Jorgensen, Evan Laske, Shaun Azimi, Maja Mataric, and Odest Chadwicke Jenkins. Comparing Human Input in RViz and Virtual Reality Interfaces for 3D Object Manipulation Tasks. https://ntrs.nasa.gov/citations/20220012814, 2022.

[247] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[248] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot Learning from Demonstration by Constructing Skill Trees. *The International Journal of Robotics Research (IJRR)*, 31(3):360–375, 2012.

[249] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. Learning and Generalization of Complex Tasks from Unstructured Demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IR)S*, pages 5239–5246. IEEE, 2012.

[250] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. Personal Robot Training via Natural-Language Instructions. *IEEE Intelligent Systems*, 16(3):38–45, 2001.

[251] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, and Ewan Klein. Mobile Robot Programming Using Natural Language. *Robotics and Autonomous Systems*, 38(3-4):171–181, 2002.

[252] Terry Winograd. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. *MIT Technical Report*, 1971.

[253] Terry Winograd. SHRDLU. https://hci.stanford.edu/winograd/shrdlu/, 2001.

[254] Terry Winograd. Understanding Natural Language. *Cognitive Psychology*, 3(1):1–191, 1972.

[255] Terry Winograd. *Language as a Cognitive Process: Volume 1: Syntax*. Addison-Wesley Pub. Co., Reading, MA, 1983.

[256] Terry Winograd. A Language/Action Perspective on the Design of Cooperative Work. In *Proceedings of the 1986 ACM Conference on Computer-Supported Cooperative Work*, pages 203–220, 1986.

[257] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition: A New Foundation for Design*, volume 335. Ablex Publishing Corporation Norwood, NJ, 1986.

[258] Matthew Marge, Carol Espy-Wilson, Nigel G Ward, Abeer Alwan, Yoav Artzi, Mohit Bansal, Gil Blankenship, Joyce Chai, Hal Daumé III, Debadeepta Dey, Mary Harper, Thomas Howard, Casey Kennington, Ivana Kruijff-Korbayová,

Dinesh Manocha, Cynthia Matuszek, Ross Mead, Raymond Mooney, Roger K. Moore, Mari Ostendorf, Heather Pon-Barry, Alexander I. Rudnicky, Matthias Scheutz, Robert St. Amant, Tong Sun, Stefanie Tellex, David Traum, and Zhou Yu. Spoken Language Interaction with Robots: Recommendations for Future Research. *Computer Speech & Language*, 71, 2022.

[259] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots That Use Language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

[260] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to Do and How to Do It: Translating Natural Language Directives into Temporal and Dynamic Logic Representation for Goal Management and Action Execution. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4163–4168. IEEE, 2009.

[261] Sonia Chernova, Jeff Orkin, and Cynthia Breazeal. Crowdsourcing HRI Through Online Multiplayer Games. In *2010 AAAI Fall Symposium Series*, 2010.

[262] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25.1, pages 1507–1514, 2011.

[263] Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. Learning from Unscripted Deictic Gesture and Language for Human-Robot Interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.1, 2014.

[264] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas M Howard. Efficient Grounding of Abstract Spatial Concepts for Natural Language Interaction with Robot Manipulators. *Robotics: Science and Systems (RSS) Foundation*, 2016.

[265] Rohan Paul, Jacob Arkin, Derya Aksaray, Nicholas Roy, and Thomas M Howard. Efficient Grounding of Abstract Spatial Concepts for Natural Language Interaction with Robot Platforms. *The International Journal of Robotics Research (IJRR)*, 37(10):1269–1299, 2018.

[266] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. LanguageRefer: Spatial-Language Model for 3D Visual Grounding. In *Conference on Robot Learning*, pages 1046–1056. PMLR, 2022.

[267] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and Where Pathways for Robotic Manipulation. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.

[268] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the Talk: Connecting Language, Knowledge, and Action in Route Instructions. *AAAI*, 2(6):4, 2006.

[269] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward Understanding Natural Language Directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

[270] Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following Directions Using Statistical Machine Translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258. IEEE, 2010.

[271] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to Parse Natural Language Commands to a Robot Control System. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, pages 403–415. Springer, 2013.

[272] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Grounding Verbs of Motion in Natural Language Commands to Robots. In *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, pages 31–47. Springer, 2014.

[273] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. A Natural Language Planner Interface for Mobile Manipulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. IEEE, 2014.

[274] Sachithra Hemachandra, Felix Duvallet, Thomas M Howard, Nicholas Roy, Anthony Stentz, and Matthew R Walter. Learning Models for Following Natural Language Directions in Unknown Environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5608–5615. IEEE, 2015.

[275] Felix Duvallet, Matthew R Walter, Thomas Howard, Sachithra Hemachandra, Jean Oh, Seth Teller, Nicholas Roy, and Anthony Stentz. Inferring Maps and Behaviors from Natural Language Instructions. In *Experimental Robotics: The 14th International Symposium on Experimental Robotics*, pages 373–388. Springer, 2016.

[276] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[277] IBM. CIMON: IBM Watson in Space. https://www.ibm.com/thought-leadership/smart/ai-in-space-xp/journal.html, 2018.

[278] Till Eisenberg. CIMON, the AI-Powered Robot, Launches a New Era in Space Travel. https://www.ibm.com/blog/cimon-ai-robot-launches-new-era-space-travel/, 2019.

[279] IBM. CIMON brings AI to the International Space Station. https://www.ibm.com/thought-leadership/innovation-explanations/cimon-ai-in-space, 2019.

[280] Martin Porcheron, Joel E Fischer, Stuart Reeves, and Sarah Sharples. Voice Interfaces in Everyday Life. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[281] Benjamin R Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. "What Can I Help you With?": Infrequent Users' Experiences of Intelligent Personal Assistants. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–12, 2017.

[282] Ewa Luger and Abigail Sellen. "Like Having a Really Bad PA": The Gulf Between User Expectation and Experience of Conversational Agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5286–5297, 2016.

[283] Jesse Mu and Advait Sarkar. Do We Need Natural Language? Exploring Restricted Language Interfaces for Complex Domains. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2019.

[284] Alexandre Antunes, Lorenzo Jamone, Giovanni Saponaro, Alexandre Bernardino, and Rodrigo Ventura. From Human Instructions to Robot Actions: Formulation of Goals, Affordances and Probabilistic Planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5449–5454. IEEE, 2016.

[285] Kai-yuh Hsiao, Nikolaos Mavridis, and Deb Roy. Coupling Perception and Simulation: Steps Towards Conversational Robotics. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 928–933. IEEE, 2003.

[286] Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis. Mental Imagery for a Conversational Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(3):1374–1383, 2004.

[287] Lanbo She, Yu Cheng, Joyce Y Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. Teaching Robots New Actions Through Natural Language Instructions. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873. IEEE, 2014.

[288] Josef Ruppenhofer, Michael Ellsworth, Myriam Schwarzer-Petruck, Christopher R Johnson, and Jan Scheffczyk. FrameNet II: Extended Theory and Practice. Technical report, International Computer Science Institute, 2016.

[289] Diana Trandabat and Dan Cristea. *Natural Language Processing Using Semantic Frames*. PhD thesis, University “Alexandru Ioan Cuza” of Iaşi, Romania, 2010.

[290] Ye-Yi Wang, Li Deng, and Alex Acero. Semantic Frame-Based Spoken Language Understanding. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 41–91, 2011.

[291] Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. Semantic Frame Identification with Distributed Word Representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1458, 2014.

[292] Brian J Thomas and Odest Chadwicke Jenkins. RoboFrameNet: Verb-Centric Semantics for Actions in Robot Middleware. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4750–4755. IEEE, 2012.

[293] Emily Sheetz, Matthew Shannon, Cameron Kisailus, Adam Ingerman, and Shaun Azimi. Hierarchical Semantic Frames for Grounding Language in Robot Control Primitives. https://ntrs.nasa.gov/citations/20240004454, 2024.

[294] Dan Klein and Christopher D Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[295] Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. Tiered Reasoning for Intuitive Physics: Toward Verifiable Commonsense Language Understanding. *arXiv preprint arXiv:2109.04947*, 2021.

[296] RA Hirschfeld, F Aghazadeh, and RC Chapleski. Survey of Robot Safety in Industry. *International Journal of Human Factors in Manufacturing*, 3(4):369–379, 1993.

[297] Tadele Shiferaw Tadele, Theo de Vries, and Stefano Stramigioli. The Safety of Domestic Robotics: A Survey of Various Safety-Related Publications. *IEEE Robotics & Automation Magazine*, 21(3):134–142, 2014.

[298] Valeria Villani, Fabio Pini, Francesco Leali, and Cristian Secchi. Survey on Human-Robot Collaboration in Industrial Settings: Safety, Intuitive Interfaces, and Applications. *Mechatronics*, 55:248–266, 2018.

[299] Shitij Kumar, Celal Savur, and Ferat Sahin. Survey of Human-Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1):280–297, 2020.

[300] Federico Vicentini. Collaborative Robotics: A Survey. *Journal of Mechanical Design*, 143(4), 2021.

[301] Matteo Rubagotti, Inara Tusseyeva, Sara Baltabayeva, Danna Summers, and Anara Sandygulova. Perceived Safety in Physical Human-Robot Interaction—A Survey. *Robotics and Autonomous Systems*, 151, 2022.

[302] Zhihao Liu, Xinran Wang, Yijie Cai, Wenjun Xu, Quan Liu, Zude Zhou, and Duc Truong Pham. Dynamic Risk Assessment and Active Response Strategy for Industrial Human-Robot Collaboration. *Computers & Industrial Engineering*, 141, 2020.

[303] Hun-Ok Lim and Kazuo Tanie. Human Safety Mechanisms of Human-Friendly Robots: Passive Viscoelastic Trunk and Passively Movable Base. *The International Journal of Robotics Research (IJRR)*, 19(4):307–335, 2000.

[304] Darko Bozhinoski, Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, and Ivica Crnkovic. Safety for Mobile Robotic Systems: A Systematic Mapping Study from a Software Engineering Perspective. *Journal of Systems and Software*, 151:150–179, 2019.

[305] Yuhao Chen, Chizhao Yang, Yu Gu, and Boyi Hu. Influence of Mobile Robots on Human Safety Perception and System Productivity in Wholesale and Retail Trade Environments: A Pilot Study. *IEEE Transactions on Human-Machine Systems*, 52(4):624–635, 2022.

[306] Robert Bogue. Robots that Interact with Humans: A Review of Safety Technologies and Standards. *Industrial Robot: An International Journal*, 44(4):395–400, 2017.

[307] Paul A Schulte, Jessica MK Streit, Fatima Sheriff, George Delclos, Sarah A Felknor, Sara L Tamers, Sherry Fendinger, James Grosch, and Robert Sala. Potential Scenarios and Hazards in the Work of the Future: A Systematic Review of the Peer-Reviewed and Gray Literatures. *Annals of Work Exposures and Health*, 64(8):786–816, 2020.

[308] Aude Billard and Danica Kragic. Trends and Challenges in Robot Manipulation. *Science*, 364(6446), 2019.

[309] Christopher Peace. The Risk Matrix: Uncertain Results? *Policy and Practice in Health and Safety*, 15(2):131–144, 2017.

[310] Patricia Guevara. A Guide to Understanding 5x5 Risk Assessment Matrix. https://safetyculture.com/topics/risk-assessment/5x5-risk-matrix/, 2024.

[311] Kimberly A Hambuchen. *Multi-Modal Attention and Event Binding in Humanoid Robot Using a Sensory Ego-Sphere*. PhD thesis, Vanderbilt University, 2004.

[312] Richard Alan Peters, Kimberly E Hambuchen, Kazuhiko Kawamura, and D Mitchell Wilkes. The Sensory Ego-Sphere as a Short-Term Memory for Humanoids. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 451–459, 2001.

[313] Richard Alan Peters, Kimberly A Hambuchen, and Robert E Bodenheimer. The Sensory Ego-Sphere: A Mediating Interface Between Sensors and Cognition. *Autonomous Robots*, 26(1):1–19, 2009.

[314] Herbert P Grice. Logic and Conversation. In *Speech Acts*, pages 41–58. Brill, 1975.

[315] Elinor Ochs Keenan. The Universality of Conversational Postulates. *Language in Society*, 5(1):67–80, 1976.

[316] Paul Grice. *Studies in the Way of Words*. Harvard University Press, 1991.

[317] Stephen Neale. Paul Grice and the Philosophy of Language. *Linguistics and Philosophy*, pages 509–559, 1992.

[318] J Thomas. Conversational Maxims. *Concise Encyclopedia of Philosophy of Language*, pages 517–518, 1997.

[319] Nicholas Allott. Paul Grice, Reasoning and Pragmatics. *UCL Working Papers in Linguistics*, 17:217–243, 2005.

[320] J. L. Meriam and L. G. Kraige. *Engineering Mechanics: Dynamics*. John Wiley & Sons, 2012.

[321] Sarah K Hopko and Ranjana K Mehta. Trust in Shared-Space Collaborative Robots: Shedding Light on the Human Brain. *Human Factors*, 66(2):490–509, 2024.

[322] Thomas B Sheridan. Human-Robot Interaction: Status and Challenges. *Human Factors*, 58(4):525–532, 2016.

[323] Lanbo She, Yunyi Jia, Ning Xi, and Joyce Y Chai. Exception Handling for Natural Language Control of Robots. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI) Extended Abstracts*, pages 139–140, 2015.

[324] Jiafan Zhang and Wei Song. Physics-of-Failure Based Model for Industrial Robot Reliability Prediction. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 729–734. IEEE, 2020.

[325] Benjamin Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000.

[326] Patrick Beeson, Joseph Modayil, and Benjamin Kuipers. Factoring the Mapping Problem: Mobile Robot Map-Building in the Hybrid Spatial Semantic Hierarchy. *The International Journal of Robotics Research (IJRR)*, 29(4):428–459, 2010.

[327] Roland Philippsen, Negin Nejati, and Luis Sentis. Bridging the Gap Between Semantic Planning and Continuous Control for Mobile Manipulation Using a Graph-Based World Representation. In *Workshop Hybrid Control Auton. Syst., Pasadena, CA*, 2009.

[328] Patrick Beeson, David Kortenkamp, R Peter Bonasso, Andreas Persson, Amy Loutfi, and Jonathan P Bona. An Ontology-Based Symbol Grounding System for Human-Robot Interaction. In *2014 AAAI Fall Symposium Series*, 2014.

[329] Leon Ziegler. *The Attentive Robot Companion: Learning Spatial Information from Observation and Verbal Interaction*. PhD thesis, Bielefeld University, 2015.

[330] Judea Pearl. Causal Diagrams for Empirical Research. *Biometrika*, 82(4):669–688, 1995.

[331] Judea Pearl. *Causality*. Cambridge University Press, 2009.

[332] Judea Pearl. The Do-Calculus Revisited. *arXiv preprint arXiv:1210.4852*, 2012.

[333] Neal J Roese. Counterfactual Thinking. *Psychological Bulletin*, 121(1):133, 1997.

[334] Neal J Roese and Kai Epstude. The Functional Theory of Counterfactual Thinking: New Evidence, New Challenges, New Insights. In *Advances in Experimental Social Psychology*, volume 56, pages 1–79. Elsevier, 2017.

[335] Kai Epstude and Neal J Roese. The Functional Theory of Counterfactual Thinking. *Personality and Social Psychology Review*, 12(2):168–192, 2008.

[336] Kai Epstude and Neal J Roese. When Goal Pursuit Fails: The Functions of Counterfactual Thought in Intention Formation. *Social Psychology*, 2011.

[337] Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal Reasoning from Meta-Reinforcement Learning. *arXiv preprint arXiv:1901.08162*, 2019.

[338] Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the Complexity of Solving Markov Decision Problems. *arXiv preprint arXiv:1302.4971*, 2013.

[339] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Third Edition*. Pearson Education, 2010.

[340] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Fourth Edition*. Pearson Education, 2020.

[341] Yusra Alkhazraji, Matthias Frorath, Markus Grützner, Malte Helmert, Thomas Liebetraut, Robert Mattmüller, Manuela Ortlieb, Jendrik Seipp, Tobias Springenberg, Philip Stahl, and Jan Wülfing. Pyperplan. https://doi.org/10.5281/zenodo.3700819, 2020.

[342] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-GraspNet: Efficient 6-DOF Grasp Generation in Cluttered Scenes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444. IEEE, 2021.

[343] Minas Liarokapis and Aaron M Dollar. Combining Analytical Modeling and Learning to Simplify Dexterous Manipulation with Adaptive Robot Hands. *IEEE Transactions on Automation Science and Engineering*, 16(3):1361–1372, 2018.

[344] Aaron M Dollar and Robert D Howe. Towards Grasping in Unstructured Environments: Optimization of Grasper Compliance and Configuration. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 4, pages 3410–3416. IEEE, 2003.

[345] Aaron Michael Dollar. *Design Principles for Robust Grasping in Unstructured Environments*. PhD thesis, Harvard University, 2006.

[346] Miquel Oller, Dmitry Berenson, and Nima Fazeli. TactileVAD: Geometric Aliasing-Aware Dynamics for High-Resolution Tactile Control. In *7th Annual Conference on Robot Learning*, 2023.

[347] Andrea Sipos and Nima Fazeli. Multiscope: Disambiguating In-Hand Object Poses with Proprioception and Tactile Feedback. *arXiv preprint arXiv:2305.14204*, 2023.

[348] Miquel Oller, Dmitry Berenson, and Nima Fazeli. Tactile-Driven Non-Prehensile Object Manipulation via Extrinsic Contact Mode Control. *arXiv preprint arXiv:2405.18214*, 2024.

[349] An Dang, James Lorenz, Xili Yi, and Nima Fazeli. Bimanual In-hand Manipulation using Dual Limit Surfaces. *arXiv preprint arXiv:2409.14698*, 2024.

[350] Nima Fazeli, Miquel Oller, Jiajun Wu, Zheng Wu, Joshua B Tenenbaum, and Alberto Rodriguez. See, Feel, Act: Hierarchical Learning for Complex Manipulation Skills with Multisensory Fusion. *Science Robotics*, 4(26), 2019.

[351] Yizhou Chen, Andrea Sipos, Mark Van der Merwe, and Nima Fazeli. Visuo-Tactile Transformers for Manipulation. *arXiv preprint arXiv:2210.00121*, 2022.

[352] Mark Van der Merwe, Dmitry Berenson, and Nima Fazeli. Learning the Dynamics of Compliant Tool-Environment Interaction for Visuo-Tactile Contact Servoing. In *Conference on Robot Learning*, pages 2052–2061. PMLR, 2023.

[353] Aabha A. Anekar and Bruno Bordoni. *Palmar Grasp Reflex*. StatPearls Publishing, Treasure Island (FL), 2020.

[354] Yasuyuki Futagi, Yasuhisa Toribe, and Yasuhiro Suzuki. The Grasp Reflex and Moro Reflex in Infants: Hierarchy of Primitive Reflex Responses. *International Journal of Pediatrics*, 2012(1), 2012.

[355] Marcus Vinicius Marques-de Moraes, Jadiane Dionisio, Uner Tan, and Eloisa Tudella. Palmar Grasp Reflex in Human Newborns. *Pediatrics & Therapeutics*, 7(309), 2017.

[356] Zachary Ray and Erik D Engeberg. Human-Inspired Reflex to Autonomously Prevent Slip of Grasped Objects Rotated with a Prosthetic Hand. *Journal of Healthcare Engineering*, 2018, 2018.

[357] Shotaro Okajima, Fady S Alnajjar, Hiroshi Yamasaki, Matti Itkonen, Álvaro Costa García, Yasuhisa Hasegawa, and Shingo Shimoda. Grasp-Training Robot to Activate Neural Control Loop for Reflex and Experimental Verification. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1849–1854. IEEE, 2018.

[358] Peter Deckers, Aaron M Dollar, and Robert D Howe. Guiding Grasping with Proprioception and Markov Models. In *Proceedings of the Workshop on Robot Manipulation: Sensing and Adapting to the Real World, Robotics: Science and Systems (RSS) Conference,(Atlanta, GA)*, 2007.

[359] PSYONIC. The World's First Touch-Sensing Bionic Hand. https://www.psyonic.io/, 2024.

[360] PSYONIC. Meet the Ability Hand. https://www.psyonic.io/ability-hand, 2024.

[361] Steven Jens Jorgensen, Michael W Lanighan, Sylvain S Bertrand, Andrew Watson, Joseph S Altemus, R Scott Askew, Lyndon Bridgwater, Beau Domingue, Charlie Kendrick, Jason Lee, Mark Paterson, Jairo Sanchez, Patrick Beeson, Seth Gee, Stephen Hart, Ana Huaman Quispe, Robert Griffin, Inho Lee, Stephen McCrory, Luis Sentis, Jerry Pratt, and Joshua S Mehling. Deploying the NASA Valkyrie Humanoid for IED Response: An Initial Approach and Evaluation Summary. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2019.

[362] Open Robotics. ROS Noetic Ninjemys. https://wiki.ros.org/noetic, 2020.

[363] Sachin Chitta, Ioan Sucan, and Steve Cousins. MoveIt! ROS Topics. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[364] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the Barrier to Entry of Complex Robotic Software: A MoveIt! Case Study. *arXiv preprint arXiv:1404.3785*, 2014.

[365] Sylvain Bertrand, Duncan Calvert, Stephen McCrory, Robert Griffin, Bhavyansh Mishra, James Foster, Dexton Anderson, Luigi Penco, and Nick Kitchel. IHMC Open Robotics Software. https://github.com/ihmcrobotics/ihmc-open-robotics-software, 2023.

[366] Open Robotics. XML Robot Description Format (URDF). https://wiki.ros.org/urdf/XML/model, 2023.

[367] Open Robotics. URDF. https://wiki.ros.org/urdf, 2023.

[368] Open Robotics. URDF Tutorials. https://wiki.ros.org/urdf/Tutorials, 2016.

[369] Kapil Dev Sharma and Shobhit Srivastava. Failure Mode and Effect Analysis (FMEA) Implementation: A Literature Review. *Journal of Advance Research in Aeronautics and Space Science*, 5(1):1–17, 2018.

[370] Peter L Goddard. Software FMEA Techniques. In *Annual Reliability and Maintainability Symposium. 2000 Proceedings. International Symposium on Product Quality and Integrity*, pages 118–123. IEEE, 2000.

[371] University of Texas School of Public Health. Failure Mode Effects Analysis (FMEA). https://www.slideserve.com/primo/failure-mode-effects-analysis-fmea-2601014, 2014.

[372] NASA Goddard Space Flight Center. Guideline for Failure Modes and Effects Analysis and Risk Assessment. https://standards.nasa.gov/sites/default/files/standards/GSFC/Baseline/0/GSFC-HDBK-8004_Approved_1.pdf, 2024.

[373] NASA Glenn Research Center. Hazards Analysis and Failure Modes and Effects Criticality Analysis (FMECA) of Four Concept Vehicle Propulsion Systems. https://rotorcraft.arc.nasa.gov/Publications/files/CR-2019-220217.pdf, 2019.

[374] Katherine B Percarpio, B Vince Watts, and William B Weeks. The Effectiveness of Root Cause Analysis: What Does the Literature Tell Us? *The Joint Commission Journal on Quality and Patient Safety*, 34(7):391–398, 2008.

[375] Occupational Safety and Health Administration. The Importance of Root Cause Analysis During Incident Investigation. https://www.osha.gov/sites/default/files/publications/OSHA3895.pdf, 2016.

[376] Jessica Ji. What Does AI Red-Teaming Actually Mean? https://cset.georgetown.edu/article/what-does-ai-red-teaming-actually-mean/, 2023.

[377] Jessica Ji. How to Improve AI Red-Teaming: Challenges and Recommendations. https://cset.georgetown.edu/article/how-to-improve-ai-red-teaming-challenges-and-recommendations/, 2025.

[378] Center for Security and Emerging Technology (CSET). What's Next for AI Red Teaming? https://www.youtube.com/watch?v=gDnNuxpvPis, 2025.

[379] Natasha Bajema. Why are Large AI Models Being Red Teamed? https://spectrum.ieee.org/red-team-ai-llms, 2024.

[380] Jessica Ji and Colin Shea-Blymyer. Revising AI Red-Teaming. https://cset.georgetown.edu/article/revisiting-ai-red-teaming/, 2024.

[381] Minae Kwon, Hengyuan Hu, Vivek Myers, Siddharth Karamcheti, Anca Dragan, and Dorsa Sadigh. Toward Grounded Social Reasoning. *arXiv preprint arXiv:2306.08651*, 2023.

[382] Wang Qiang and Zhan Zhongli. Reinforcement Learning Model, Algorithms and Its Application. In *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, pages 1143–1146. IEEE, 2011.

[383] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access*, 7:133653–133667, 2019.

[384] Jesse Clifton and Eric Laber. Q-Learning: Theory and Applications. *Annual Review of Statistics and Its Application*, 7:279–301, 2020.

[385] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[386] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 2013.

[387] David Silver, Alex Graves, Ioannis Antonoglou, Martin Riedmiller, Volodymyr Mnih, D Wierstra, and K Kavukcuoglu. Playing Atari with Deep Reinforcement Learning. *DeepMind Lab. arXiv*, 1312, 2013.

[388] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[389] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-Conditional Video Prediction Using Deep Networks in Atari Games. *Advances in Neural Information Processing Systems*, 28, 2015.

[390] Melrose Roderick, James MacGlashan, and Stefanie Tellex. Implementing the Deep Q-Network. *arXiv preprint arXiv:1711.07478*, 2017.

[391] Yan Chen and Elis Kulla. A Deep Q-Network with Experience Optimization (DQN-EO) for Atari's Space Invaders. In *Web, Artificial Intelligence and Network Applications: Proceedings of the Workshops of the 33rd International Conference on Advanced Information Networking and Applications (WAINA-2019) 33*, pages 351–361. Springer, 2019.

[392] Gary King and Langche Zeng. Logistic Regression in Rare Events Data. *Political Analysis*, 9(2):137–163, 2001.

[393] The Iron Giant, 1999.

[394] 2001: A Space Odyssey, 1968.

[395] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL—The Planning Domain Definition Language. *Technical Report, Tech. Rep.*, 1998.

[396] National Aeronautics and Space Administration. 5 Hazards of Human Spaceflight. https://www.nasa.gov/hrp/hazards/, 2024.

[397] David Roos. 10 Home Repairs That Can Seriously Break the Bank. https://home.howstuffworks.com/home-improvement/repair/10-home-repairs-break-bank.htm, 2014.

[398] Jessica Bennett. 9 Cleaning Mistakes That Are Making Your Home Dirtier. https://www.bhg.com/homekeeping/house-cleaning/tips/cleaning-mistakes/, 2024.

[399] Adam Fraiel. 10 Common Travel Problems and How to Deal with Them. https://www.worldpackers.com/articles/common-travel-problems-and-how-to-deal-with-them, 2025.

[400] Cherise Threewitt. 10 Serious Mistakes in Car Maintenance. https://auto.howstuffworks.com/under-the-hood/vehicle-maintenance/10-serious-mistakes-car-maintenance.htm, 2013.

[401] Centers for Disease Control and Prevention (CDC). Nuclear Weapon Infographic. https://www.cdc.gov/radiation-emergencies/infographic/nuclear-weapon.html, 2024.

[402] Rina Diane Caballar. 10 AI Dangers and Risks and How to Manage Them. https://www.ibm.com/think/insights/10-ai-dangers-and-risks-and-how-to-manage-them, 2024.

[403] Center for AI Safety. An Overview of Catastrophic AI Risks. https://www.safe.ai/ai-risk, 2024.

[404] National Aeronautics and Space Administration. Moon to Mars Strategy and Objectives. https://www.nasa.gov/moontomarsarchitecture-strategyandobjectives/, 2022.

[405] JSC Safety and Mission Assurance (SMA) Flight Safety Office. Significant Incidents and Close Calls in Human Spaceflight. https://sma.nasa.gov/SignificantIncidents/, 2024.

[406] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, Architecture, and Uses in the Wild. *Science Robotics*, 7(66), 2022.

[407] PickNik Robotics. MoveIt2 Documentation. https://moveit.picknik.ai/main/index.html, 2024.

[408] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[409] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, Zeng Yifu, Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, November 2022.

[410] Roger E Beaty. The Creative Brain. In *Cerebrum: The Dana Forum on Brain Science*, 2020.

[411] Benjamin Kuipers. Airplanes in the Sky. https://web.eecs.umich.edu/~kuipers/opinions/airplanes.html, 2016.

[412] Charles Percy Snow. The Two Cultures. https://web.archive.org/web/20170508085255/https://www.rbkc.gov.uk/pdf/Rede-lecture-2-cultures.pdf, 1959.

[413] John McCarthy. Programs with Common Sense. In M. L. Minsky, editor, *Semantic Information Processing*, pages 403–418. MITP, Cambridge, MA, 1968.

[414] Hardik Parwana. *Online Adaptation for Safe Control of Constrained Dynamical Systems*. PhD thesis, University of Michigan, 2024.