Abstract

Fusing Symbolic and Subsymbolic Approaches for Natural and Effective Human-Robot Collaboration

Jake Brawer

2023

Human-robot collaboration (HRC) is a field that studies how to combine the strengths of humans and robots to perform joint tasks. A crucial element of any successful collaboration is the ability for collaborators to flexibly adapt to each others' needs. In the context of HRC, this not only implies that robots can modify their behavior in-the-moment to directives issued by human users, but also that they can rapidly acquire the skills necessary for the task at hand. In turn, the knowledge acquired by a collaborative robot should be transparent and easily accessible, allowing users to adapt their behavior based on the robot's capabilities and limitations. Ideally, we could design these systems using the remarkably powerful, data-driven tools developed by the machine learning community such as deep learning. However, the black-box, *subsymbolic*, nature of many of these techniques means that they lack the requisite adaptability and transparency for HRC. Traditional *symbolic* reasoning systems, in contrast, tend to produce easily interpretable and adaptable systems, however, they often lack the scalability and flexibility offered by machine learning methods. We believe that the ideal HRC framework exists at the intersection of these symbolic and subsymbolic traditions.

In this thesis, we describe methods for leveraging symbolic and subsymbolic knowledge approaches for improving the naturalness, fluency, and flexibility of HRC. Given the importance of manufacturing as an application domain for HRC, we first focus on improving a robot's ability to use and reason about tools. In particular, we present a method by which a robot can learn symbolically instantiated cause-and-effect re-

lations underlying tool use via self-supervised experimentation. The robot then uses these relations to construct tool affordance models, enabling it to effectively complete goal-directed tool-use tasks. Subsequently, we demonstrate how such models can be used to guide and improve the subsymbolic skill learning process. We not only show that our approach can quickly learn a wide variety of skills, but readily transfer them to novel tools and manipulated objects without additional training.

We also demonstrate how symbolic abstractions can improve the social components of HRCs. We show how symbolic models of context can augment the capabilities of language models to interpret naturalistic and situated user commands. We also demonstrate how predicate logic rules can act as a powerful interface between a user's communicated intentions for a collaborative robot and the robot's behavior. We show this first in the context of ownership norm learning. We demonstrate a method whereby3 logically encoded ownership norms can be used to constrain the robot's behavior as well as to guide the inference of ownership relations of objects in a shared workspace. Finally, we develop a generalized framework for grounding user directives for a robot to mutable and composable logical rules. These rules then act as constraints on a robot's reinforcement learning policy, enabling a user to immediately modify the robot's behavior to their own ends. The work presented in this thesis contributes to the goal of creating intelligent, responsive, and capable robot collaborators.

Fusing Symbolic and Subsymbolic Approaches for Natural and Effective Human-Robot Collaboration

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Jake Brawer

Dissertation Director: Brian Scassellati

February, 2023

Contents

List of Figures

List of Tables

Acknowledgment

I am eternally grateful to those who have dragged me, often kicking and screaming, across the finish line of my Ph.D.

First, I'd like to thank my Ph.D. advisor Brian Scassellati. In 2015, Scaz gave a talk at Vassar College that changed my life. Since then he has helped me become a better critical thinker, effective communicator and independent researcher. It has been the honor of a lifetime to work with him and I am so, so grateful to have been given this opportunity.

Of course many others at Yale were absolutely instrumental in getting me to this point. I am indebted to Alessandro Roncone and Olivier Mangin. They provided invaluable guidance early on, and were never afraid to give me the (delightfully European) kick-in-the-butt I needed (and often still need). More recently, Marynel Vázquez has provided absolutely essential support to my research endeavors; I am grateful as much for her wisdom as her seemingly endless positivity.

I am extremely thankful for my lab-mates, past an present, for enriching my time at Yale immensely: Elena Corina Grigore, Aditi Ramachandran, Sarah Sebo, Meiying Qin, Nicole Salomons, Timothy Adamson, Emmanuel Adeniran, Rebecca Ramnauth, Nicholas Georgiou, Debasmita Ghose, Kate Candon, Kayla Matheus, Ellie Mamantov, Nathan Tsoi, Sydney Thompson, Austin Narcomey, Qiping Zhang, Kate Tsui, Chien-Ming Huang, and Laura Boccanfuso. In particular, I'd like thank Sarah, without whom I may have never escaped the desert. I am especially indebted

to those in the lab with whom I had the opportunity to collaborate. Indeed, much of the work in this thesis would not have been possible without Meiying; thank you for all of your hard work and dedication. Similarly, I could not have asked for better collaborators on the chefbot project than Debasmita and Kate.

I would be remiss if I did not mention my truly amazing friends. My deepest thanks goes to the KBT Crew, the Couch Crew, and the Lo-Fi Boys. To most, these names mean absolutely nothing, but to me they mean a great deal. Chris, Alex, and Henry, you made my time here an absolute joy. Because of you, I can say without the slightest hint of irony that the true thesis really is the friends I made along the way.

Words will no doubt fail to capture my gratitude for my family, though I will attempt it any way. Mom, Dad, Max: thank you so much for all your support. I literally would not be here without you, and I wouldn't have it any other way.

Last but not least, Sybil, my heart, and our little cat Edward: Thank you so much. Sybil, my love and sincerest thanks for the too-many times you've picked me up when the work in this thesis has brought me to my knees. I mean it when I say I could not have done any of this without you. Most of all, I am grateful that I always have you to remind me to dance so it all keeps spinning.

Everything was beautiful, and nothing hurt.

KURT VONNEGUT

Chapter 1

Introduction

Human-robot collaboration (HRC) is a sub-field of human-robot interaction (HRI) concerned with leveraging the strengths of both humans and robots in order to complete a joint task, often in the domains of furniture construction (Roncone et al., 2017) or other factory-related applications (Hayes and Scassellati, 2015; Tellex et al., 2014b). In order to be effective collaborators, robots should be able to interact in a manner that is most comfortable to humans, i.e., in a way that while primarily linguistic, is also multi-modal (Wahn et al., 2016) and contextual (Shah and Breazeal, 2010). Central to these capabilities is the notion of knowledge transparency, which not only can also enable robots to explain their behavior in a clear way (Hayes and Shah, 2017a), but is crucial to allowing humans to "predict the robot's behavior based on ascribed beliefs, intentions and desires" (Scheutz et al., 2007).

In addition to the need for robot knowledge to be transparent, we argue that it must also be quickly and easily adaptable to human intentions. This means being responsive to human directives and instructions, as well as contending with directives that are provided intermittently, are high-level or imprecise, or are merely implied. An impediment to designing a system with these capabilities is that the current trends in machine learning (ML) have broadly been moving in the opposite direction

of transparency and adaptability. ML techniques like supervised and unsupervised learning (Niekum et al., 2013; Jenkins and Matarić, 2004) and more recently deep reinforcement learning (DRL; Zhu et al., 2017) are popular and successful methods for learning robot policies. However, these systems typically rely on large data sets, are unable to offer explanations for how they work, and struggle to transfer knowledge to new contexts (Marcus, 2020), making them ill-equipped for an application domain like HRC that prioritizes learning that is fast, inspectable, easily altered, and readily generalized.

In contrast, symbolic planners represent the world using discrete, abstracted quantities which can enable clear communication, can generically apply across contexts and instances, and can be more readily modified and amended even by naive users. Additionally, as language is itself a symbolic medium, a system capable of symbolic reasoning can more readily incorporate directives communicated via natural language into its action plans. On the other hand, these approaches typically rely on hand-coded knowledge, rather than knowledge extracted automatically from data, and so cannot scale as readily as ML approaches.

As a consequence of this, many researchers have sought to combine *symbolic knowledge* approaches with ML, or *subsymbolic knowledge* approaches, in the hopes complimenting strengths of each while minimizing their weaknesses. More specifically, we have identified three main trends across the literature motivating these sorts of combined architectures. These include the development of i) *more powerful human-robot interfaces* which entail not only the ability to understand and integrate human-produced cues, but to be understood by people via the ability to represent knowledge in a transparent way; ii) *better knowledge generalizability and transfer*; iii) *faster and safer* learning and planning. In the following sections, we will delve into each of these motivations in more detail, exploring the ways in which researchers have sought to address these challenges through the use of hybrid symbolic-subsymbolic

architectures.

1.1 More Powerful Human-Robot Interfaces

In a collaborative setting, good communication can be the difference between task success and failure. In the medical field, for example, it has been demonstrated that standardizing communication protocols between healthcare providers can have a drastic effect on patient care (Pfrimmer, 2009; Leonard et al., 2004). Likewise, research in HRI and human-computer interaction (HCI) has shown that systems with transparent decision-making processes assist users in calibrating their levels of trust in the system (Lee and Moray, 1992; Wang et al., 2009), as well as how they attribute blame to it (**?**). Therefore, enabling robots to communicate effectively with humans is vitally important for advancing the field of HRC. This means not only enabling robots to understand and act on human commands but to represent and explain the contents of their knowledge back to humans.

1.1.1 Natural Language Understanding

The ability to interpret user natural language speech, sometimes referred to as natural language understanding (NLU) is a well-studied problem, and one at which combined subsymbolic and symbolic approaches have excelled. Given the relatively recent successes of language models trained using transformers (a neurosymbolic architecture see Sec. 2.2.1), these models have quickly become a powerful tool for robotics applications. Such language models have been proposed as a means to augment the knowledge base of cognitive architectures like SOAR (Wray et al., 2021), but also a means of mapping natural language instructions to actions (Storks et al., 2021; Singh et al., 2022; Guhur et al., 2022; Stengel-Eskin et al., 2022a; Ahn et al., 2022).

Another central capability of NLU for robotics is for robots to be able to learn new

concepts and words. In order to do this, these systems must be able to solve the so-called symbol-grounding problem (Vogt, 2002), that is, the problem of how symbols (namely words) come to be reliably associated with things in the world. From an implementational standpoint, the name "symbol-grounding problem" alone obviates the need for a hybrid architecture of the sort described above; such a system must not only be capable of producing and manipulating symbols, but learning associations in a manner at which subsymbolic approaches excel. Indeed, this observation is largely born out in the literature; The DIARC cognitive architecture (Scheutz et al., 2019) is able to learn novel object parts in one-shot (Scheutz et al., 2017) such as "knife handle" by transforming the concept into a series of relational predicates, which when combined with extant knowledge about knives is able to refine statistically-mediated visual search to the knife handle in question. This use of logical predicates acting as an intermediate abstraction between natural language and low-level sensorimotor data is common (Wächter et al., 2018; Alomari et al., 2017; Whitney et al., 2016). Indeed, in Chapter 7 we demonstrate how high-level user-provided directives can be used to quickly modify a learned robot policy by means of such an intervening symbolic interface. In addition to predicates, probabilistic graphical models have been employed as an intermediate mapping layer (Dindo and Zambuto, 2010; Takano and Nakamura, 2012; Patki et al., 2019). In general, grounding language in the real world can yield many practical benefits. In Chapter 5 we demonstrate this empirically by showing that NLU of even highly semantically ambiguous task-based utterances can be drastically improved when grounded to learned task models.

1.1.2 Explainability

Another important aspect for successful human-robot communication is the ability for robots to explain and represent the contents of their knowledge explicitly. This has been a challenge with the rise of deep learning, as the black-box nature of these

systems do not lend themselves to easy inspection (Marcus, 2018). In response to this, the field of eXplainable AI (XAI) has emerged (Samek et al., 2017), often employing many of the combined symbolic-subsymbolic techniques described throughout this dissertation. In terms of integrative approaches, as with NLU, transformer models and graph neural networks (GNNs) are becoming a popular option given their ability to produce and manipulate comprehensible text and graph representations. For example, the ability to provide natural language captions of images has been demonstrated as a means of explaining the visual input to a domestic mobile robot (Cornia et al., 2020). GNNs have also been employed to produce interpretable graph representation of autonomous vehicle behavior during various multi-vehicle (Tang et al., 2021) or traffic scenarios (Limeros et al., 2022).

Nevertheless, as with many NLU approaches, XAI approaches in robotics tend to opt for the techniques that utilize some intermediate symbolic interface between a statistically-learned policy and the robot's sensors, in this case usually some sort of text or speech generator. Often this interface is based on variants of predicate or relational logic (e.g., Hayes and Scassellati, 2016; Wächter et al., 2018; Gopalan et al., 2018; Berg et al., 2020), as such representations can concisely and meaningfully represent the low-level observation space policies usually reason over. For example, one approach enables naive users to query a robot to explain particular actions, by first producing a symbolic representation of the actions' preconditions using grounded high-level predicates grounded to the robot's state space (Hayes and Shah, 2017a). This basic idea has also been applied to reward function decomposition (Iucci et al., 2021), that is, symbolically representing the criteria for certain RL rewards, as a means of providing an explanation for certain behaviors. A similar approach (Verma et al., 2021) answers user queries about blackbox robot policies by first attempting to transform that policy into equivalent relational logic action-rules inspired by STRIPS (Fikes and Nilsson, 1971).

1.2 Better Knowledge Transfer and Generalization

While representing knowledge in a symbolic way can render a robot's behavior more transparent to users, symbolic abstractions can enable a robot to more readily generalize and transfer its knowledge to novel scenarios. This is crucial in a robotics context as data needed to train new policies can be difficult to acquire due to the limitations that come with physical embodiment. Symbolic knowledge is able to generalize well because, like any good abstraction, it captures only the essential features of some phenomenon, rendering it applicable across its different instantiations (refer to Sec. 2.1 in Chapter 2 for a discussion of abstractions). Cause and effect relationships are a good example of this as symbolic abstractions like structural causal models can be used to reason about the behavior of a system under various conditions (Pearl, 2000). In some sense, robots are well-situated to learn such abstractions, as it has been shown that under most conditions specifying these relationships fully require interventional data (Pearl, 2000). Indeed causal models have been utilized across robotics including calculating efficient motion plans (Erdem et al., 2011), social navigation (Stocking et al., 2022), detecting functional similarities between different tools (Sinapov and Stoytchev, 2008), adapting robot policies to user preferences (Angelov et al., 2019), and learning how to fold clothing (Xiong et al., 2016) to name a few. In Chapter 3, we describe a method by which a robot can learn causal models via self-directed experimentation, and then use these models to construct tool affordance models to reason about familiar and novel tools.

Affordances are themselves powerful abstractions that can enable robots to reason about objects and the environment in efficient and generalizable ways. Thus much work has gone into developing methods for robots to learn affordances, in particular object (Moldovan et al., 2013; Manuelli et al., 2019; Lueddecke et al., 2019) as well as tool (Gonçalves et al., 2014a; Dehban et al., 2016; Myers et al., 2015) affordances. In Chapter 4 we show how tool affordance models can facilitate the transfer of learned

tool-use skills both to novel tools but also completely different robot platforms. While much less studied in robotics (though see Sarathy et al., 2017; Malle et al., 2017 for some examples), social affordances like social norms are crucial for robots to be able to exist effectively in human environments. To this end, in Chapter 6 we describe a method for learning and utilizing ownership norms in the context of a human-robot collaboration.

A closely related topic to causal learning is model-based RL, as the transition function being modeled by such systems can be understood as a model of the effects a robot's actions have on the environment. Such models are valuable because they describe in general terms the dynamics of the environment, which makes them generalizable both across instances of a given task as well as across multiple tasks, so long as these dynamics remain constant. For this reason, model-based RL approaches have been applied widely throughout robotics, including autonomous navigation for flying (Bagnell and Schneider, 2001; Abbeel et al., 2006) and wheeled (Shaker et al., 2009; Touzet, 1997; Martínez-Marín and Duckett, 2005) platforms, bipedal walking (Morimoto and Atkeson, 2002; Mordatch et al., 2016) and manipulation (Deisenroth et al., 2014; Van Rooijen et al., 2014; Kupcsik et al., 2017) to name a few. One drawback of such approaches is that these models are generally approximated via data-driven statistical learners like neural networks, and thus their predictive accuracy quickly diminishes over multi-time-step prediction horizons. Symbolic transition models do not share this same problem, and indeed recent work as explored efficient means of learning such models as STRIPS-style rules (Chitnis et al., 2021) and even grounding the symbols comprising these rules to raw sensorimotor data (Konidaris et al., 2018).

Much like model-based RL, HRL also involves learning abstractions to aid in making policy learning more sample-efficient and transferable. Where they differ is in the target of the abstraction process; HRL learns temporal abstractions in the form of options or skills, decomposing a task into subcomponents (see Sec. 2.2.4). Model-

based RL has been applied successfully to a wide variety of applications including object pick-and-place (Yang et al., 2021b; Wulfmeier et al., 2019), soccer shooting (Ji et al., 2022), navigation (Wöhlke et al., 2021; Chen et al., 2008; Kawano, 2013), and assembly (Hou et al., 2020).

1.3 Speed and Safety

Aside from making training data difficult to collect, a robot's physical embodiment poses two other challenges uncommon to software agents: i) acting in the world often requires generating particular and complex trajectories in the high-dimensional and continuous configuration space of the robot, and ii) unconstrained exploratory actions are not possible due to the real physical risks a robot posses to itself and others (Garcıa and Fernández, 2015). The latter challenge ii) is especially important in an HRC context which definitionally finds humans and robots interacting in a shared space. As a result, much effort has been dedicated to designing systems capable of mitigating these issues, often via a combined subsymbolic and symbolic approach.

The first challenge is the motivating factor underlying the emergence of Task and Motion Planning (TAMP) approaches (Garrett et al., 2021a; refer to Sec. 2.2.3). In TAMP, a symbolic planner provides a high-level symbolic task plan as action skeletons that are then parameterized by low-level-motion planners. This scheme is motivated by the observation that the configuration space of the robot has a modal structure contingent on the state of the objects in the world, and that constraining planning to the appropriate mode (by way of the action skeletons) can confer significant reductions in search time. As a result, TAMP has been employed to solve a wide-variety of multi-step manipulation tasks including object stacking (Toussaint, 2015), tool-use and meta-tool-use (Toussaint et al., 2018), household chores (Pack and Lozano-Pérez, 2011; Garrett et al., 2018), and fork-lift operations (Gravot et al., 2005).

Symbolic high-level constraints are not only used to speed up motion planning, but to mitigate challenge ii), i.e., risky or dangerous behavior. This is typically accomplished via techniques like constrained RL or action shielding (see Sec. 2.2.3), which enable these systems to take advantage of state of the art machine learning techniques while highly penalizing or outright preventing unsafe actions. Logic-based shielded RL, for example has been employed as a means of creating user-friendly policy repair interfaces (van Waveren et al., 2022), to learn safe policies in service robots (Jansen et al., 2018b), and contact-rich obstacle avoidance (Thananjeyan et al., 2021).

Outside of risk mitigation or policy repair action shielding offers other opportunities for HRC, namely the possibility of real-time policy shaping from user commands. This is because these approaches act the level of a robot's policy, rather than, for example, the reward function, and so can produce immediate changes to a robot's behavior without additional retraining. In Chapter 7, we discuss our own approach for this sort of policy shaping, using composable and mutable sets of logic rules to enable a user to quickly modify a collaborative robot's behavior.

1.4 Next Steps

In an effort to improve the fluency, naturalness, and flexibility of HRC, the novel work presented in this dissertation leverages both symbolic and subymbolic knowledge. We begin in Chapter 2 with a review of proposed approaches for combining symbolic and subsymbolic knowledge. While in this chapter we provided a broad overview for the motivating these architectures, in Chapter 2 we discuss how specifically these forms of knowledge are combined. In so doing, we develop a novel taxonomy of symbolic-subymbolic approaches.

In Chapter 3, we describe a method for learning and applying the high-level causal structure underlying a tool-manipulation task. We show how this knowledge can be

leveraged to construct sophisticated and generalizable tool representations, enabling a robot to reason effectively about tool affordances and solve tool selection tasks.

The work presented in Chapter 4 builds on this insight in order to develop the TransferIng Skilled Tool use Acquired Rapidly (TRI-STAR) framework. That is, TRI-STAR leverages high-level task and affordance knowledge to guide the low-level processes governing the learning of flexible tool skills that can generalize to unseen tools.

In Chapter 5 we extend this top-down relationship to natural language understanding in a HRC manufacturing setting. In particular we show how high-level task structure knowledge can be used to significantly improve the interpretation of user commands.

Chapter 6 represents initial work seeking to merge symbolic and subsymbolic knowledge more holistically, though in the specific context of ownership norm learning. We devise an approach for a robot to interactively learn norms relating to object ownership, and use statistical methods to infer ownership relations based on perceptual features of the objects, which guide and constrain a robot's behavior during a collaboration.

Chapter 7 present the Transparent Matrix overlay framework, a generalized framework for leveraging subsymbolic and symbolic knowledge in tandem. Our framework represents user-specified directives as logical rules that act as constraints on a robots policy, enabling a user to quickly modify a robot's behavior without the need to retrain it.

We conclude this dissertation in Chapter 8 with a summary of the work presented in previous chapters, including the contributions of our work, as well the limitations and avenues for future work.

Chapter 2

A Review of Combined Subsymbolic and Symbolic AI Approaches

As we saw in the previous chapter, combining symbolic and subsymbolic knowledge approaches can yield many benefits for HRC and robotics broadly. However the ways these have been combined and the motivations for combining them can differ substantially across approaches, especially when considering the wide range of approaches relevant to robotics. In this chapter, we seek to organize and discuss the various ways symbolic and subsymbolic knowledge has been combined in order to benefit robotics. We begin first with a more thorough discussion of the differences between symbolic and subsymbolic knowledge. Subsequently, we attempt to organize combined approaches across robot-relevant domains taxonomically to facilitate easier comparison.

2.1 Symbolic and Subsymbolic Knowledge

Before we can understand how symbolic and subsymbolic knowledge are combined, it is important to first understand to what these terms refer, their corresponding strengths and weaknesses, and why we might want to combine them at all. Given that we are speaking about knowledge, cognitive science can provide an intuitive base from which we can understand these subtle concepts. In cognitive science, it has often been said that cognition is an interplay between top-down and bottom-up processes (Sun, 2004). Generally speaking, bottom-up processes take in sensory data and from them extract meaningful information, while top-down processes interpret and manipulate this information utilizing previously acquired knowledge. Bottom-up processes are often associated with implicit or procedural knowledge like motor-skills (Smolensky, 1988), that, while honed through repeated practice, are consciously inaccessible and evident only in the demonstration of a skill, such as bike-riding. Top-down processes, in contrast, are explicit, deliberate, and usually under conscious control, and are associated with analytical or conceptual reasoning (Dreyfus et al., 2000).

In terms of AI and robotics, top-down and bottom-up processes find their parallels in distinct and cloistered symbolic and subsymbolic research traditions, respectively. The symbolic research tradition is most closely associated with symbol manipulation approaches that fall under the umbrella of "Good Old-Fashioned AI" (GOFAI) (Haugeland, 1989). The defining feature of these approaches are their use of formal logic for rule-based reasoning, though this term encompasses approaches involving graph-based algorithms, formal grammar, and natural language question-answering (Sarker et al., 2021). Typically these approaches attempt to model and implement what has been called common-sense reasoning or inferential learning, which includes things like conceptualization, goal-oriented planning, causal reasoning, deduction, induction, and abduction (Wang and Li, 2016; Garcez and Lamb, 2020). In contrast, subsymbolic knowledge is associated with the function-approximation or algorithmic

learning approaches broadly identified with "machine learning". The term "subsymbolic" is meant to highlight the fact that while these systems can learn powerful representations from the underlying data, they tend to lack the explicit and readily interpretable semantics of symbols (paralleling the unconscious aspect of bottom-up cognitive processes). In recent decades neural networks and in particular deep neural networks (DNNs) have become the dominant machine learning approach, but subysmbolic approaches encompass all supervised, unsupervised, and reinforcement learning approaches including k-means clustering, support vector machines, linear and logistic regression, and decision tree classifiers. These sorts of approaches excel at tasks requiring pattern recognition and data classification.

An important distinction between symbolic and subsymbolic knowledge is reperesentational. Symbolic knowledge is often closely associated with abstraction, a type of knowledge that is said to "distill the essence from its superficial trappings" (Ram and Jones, 1994). From this definition we can extract three desiderata of abstract knowledge: 1) abstractions are transformational; they distill or re-present information in some new way, 2) this new representation captures the essence or the invariant qualities of the original information, and 3) this representation is in some way simpler or more condensed than the original. Symbols are broadly seen as good representations of abstract knowledge as they are discrete and explicit, encode precise and generally applicable knowledge, and are extremely concise, mapping quite well to the three desiderata above. For example, the symbolic inference rule $cat(x) \rightarrow hasTail(x)$ encodes invariant quality of cats that applies irrespective of particulars of the robot's perceptual experience of a cat x and indeed irrespective of any particular cat x.

Subsymbolic knowledge, in contrast, represents knowledge in a manner that is in many ways in direct opposition to symbolic knowledge, and at first, blush appears decidedly non-abstract. As Sarker et al. (2021) argues, in a standard DNN, for example, a rule such as the one posed above would not be represented as an explicit

symbolic expression, but rather as a subsymbolic embedding, or latently represented in the weighted activation distributed over many neurons. These approaches, which are programmed primarily via learning, and manipulate high-dimensional, real-valued vectors embedded in a differentiable vector space, stand in stark contrast to the discrete, non-differentiable, often hand-coded, and readily interpretable world of symbolic approaches. While DNNs offer the starkest representational contrast to symbolic approaches, this same distinction can be found in the vast majority of ML approaches; in essence, these approaches tune numerical model parameters as a function of the underlying data. This produces an implicit knowledge evident only in the performance of the model on new data rather than in the production and manipulation of explicitly learned concepts or representations.

Yet, despite these apparent differences, subsymbolic systems are capable of producing and manipulating abstractions in much the same way as symbolic systems do. As Konidaris (2019) argues, by virtue of the fact that input to DNNs are highly processed via the use of intermediate layers that are often smaller in size than the input layer, the input is transformed and represented in a condensed form in the sense of desiderata 1) and 3) outlined above. Additionally, as has been demonstrated in convolutional neural networks (Zeiler and Fergus, 2014), these layers often encode intuitively desirable properties of the underlying data such as class discrimination and compositionality, fulfilling desiderata 2).

Clear differences emerge, however, when considering the functional distinctions between symbolic and subsymbolic knowledge. Marcus (2020) discusses this distinction between what he calls symbol manipulation systems and machine learning systems. According to Marcus, symbol manipulators are composed of four essential attributes: *variables*, or ungrounded manipulable entities; *instances*, or specific potential realizations of variables, *bindings*, which ground instances to variables, and *operations* which provide a means for relating variables to one another or manipulating them.

Under this scheme, knowledge is represented as operations over variables. Such a representation is not only highly generalizable in that it can apply to all instances of the variable class by way of bindings, but are also reliable and precise in that these operations are stably defined rather than learned. In contrast, machine learning offers a completely distinct paradigm that Marcus deems function approximation. Here, inputs are mapped to outputs via a process somewhat disparagingly referred to as "curve-fitting." These systems tend to be brittle, data-intensive, and limited in their capacity to generalize outside of the training distribution. Pearl (2018a) extends this criticism of such approaches by arguing that they lack the crucial ability to represent causal relationships. As a consequence, the knowledge they acquire can only ever be associational, putting an inherent limitation on their reasoning abilities. However, proponents of these approaches cite the end-to-end nature of their learning, free of the need for expert knowledge engineering to be a strength of these approaches rather than a weakness. Indeed, the spectacular real-world successes of generative models like DALLE-2 (Marcus et al., 2022), GPT-3 (Floridi and Chiriatti, 2020), PaLM (Chowdhery et al., 2022), and SayCan (Ahn et al., 2022) provide compelling and tangible evidence to this claim.

Nevertheless, there is an opportunity here to combine symbolic and subsymbolic knowledge in some way. That is, the strengths of symbolic systems – their expressiveness and ability to generalize outside of the training distribution – almost perfectly complement the weakness of subsymbolic systems – their opaqueness and unconvincing generalization capabilities. Similarly, the impressive data-driven learning abilities of statistical systems and the relative ease with which these systems can scale contrasts with the relatively hand-engineered and thus difficult-to-scale nature of symbolic systems. Ideally, an approach that combines subsymbolic and symbolic can leverage their respective strengths in order to minimize their respective weaknesses. From this, though, a question emerges: *how specifically should these forms of knowledge*

be combined? In the following section, we provide a broad overview of the different approaches for combining these forms of knowledge and organize them taxonomically.

2.2 Combining Subsymbolic and Symbolic Knowledge

In this section, we will discuss our novel taxonomic organization of robot-relevant combined subsymbolic and symbolic approaches. Developing such a taxonomy challenging because robotics research encompasses an enormous diversity of focuses including perception, natural language understanding, and generation, multi-step reasoning and planning, and motion planning. Each of these areas presents its own unique challenges and requires a specific set of approaches to address them. As a consequence, a similarly diverse array of approaches have been developed leveraging both symbolic and subsymbolic knowledge. These include a multitude of methods across different variants of reinforcement learning (RL) including model-based RL (MRL; Bagnell and Schneider, 2001; Martínez-Marín and Duckett, 2005; Mordatch et al., 2016; Kupcsik et al., 2017) and hierarchical RL (HRL; Sutton et al., 1999; Bacon et al., 2017; Dietterich, 2000; Nachum et al., 2018), Integrated Task and Motion Planning (TAMP; Garrett et al., 2021b, 2018; Hasan et al., 2020), causal reasoning (Heckerman et al., 1997; Spirtes et al., 2000a; Hoyer et al., 2008), and many cognitive architectures (Anderson et al., 1997; Laird et al., 1987; Scheutz et al., 2019; Trafton et al., 2013).

In particular, neurosymbolic AI (NeSy) approaches (Garcez and Lamb, 2020; Sarker et al., 2021; Kautz, 2022), which seek to unify the symbolic GOFAI and artificial neural network (ANN) traditions have emerged in recent years as a popular area of inquiry and a good starting point for understanding and categorizing hybrid approaches more broadly. Explicit calls to unify the top-down symbolic approaches

with bottom-up connectionist approaches date back to at least the early 1990s (Minsky, 1991), though it's been argued (Sarker et al., 2021) that this field predates the field of AI itself with the landmark paper of McCulloch and Pitts (1943).

Despite this storied history, there remains no consensus on precisely how these traditions should be integrated in a practical sense. Indeed, there have been at least two attempts to taxonomize the various NeSy approaches (Kautz, 2022; Bader and Hitzler, 2005). Important for our purpose is a representational distinction implicit in Kautz (2022) but explicit in Bader and Hitzler (2005) is between so-called *hybrid* and *integrative* approaches. Hybrid architectures are modular, allowing symbolic and subsymbolic approaches to be applied separately, though their outputs may serve as inputs for one another. AlphaGo (Wang et al., 2016a) is cited as an example of such an approach as it uses DNNs to evaluate candidate moves suggested by a distinct and decidedly GOFAI game tree search module (Kautz, 2022). Integrative systems in contrast are monolithic in the sense that the abstractions are integrated across the weights of the network itself and so cannot be easily teased apart from statistical learning. An example of an integrative system is the increasingly popular graph neural networks (GNNs; Scarselli et al., 2008) which extend the capabilities of deep neural networks (DNNs) to perform the sorts of graph-based inference hitherto relegated to the symbolic AI traditions.

Beyond this dichotomy, however, these proposed taxonomies focus on approaches specifically employing ANNs. For example, while Bader and Hitzler (2005) explicitly acknowledge hybrid approaches, the remaining seven dimensions of the taxonomy, e.g., if a NeSy approach employs "connectionist" or "neuronal" networks, are ANN-specific. As a result, they do not easily apply to all robot-relevant domains like TAMP that do not typically employ ANNs at all, or approaches in HRL and MRL where ANNs may be employed, but not necessarily so. Therefore, in order to organize and facilitate comparison between robot-relevant combined symbolic and subsym-

bolic approaches, we propose our own taxonomy depicted in Fig. 2.1. Our taxonomy recognizes the distinction between hybrid and integrative approaches to be fundamental. Additionally, given the inherent separability of symbolic and subsymbolic knowledge within hybrid approaches, we can further characterize approaches based on the manner in which these types of knowledge are combined. With *top-down hybrids*, knowledge is combined unidirectional from the abstract to the statistical usually by means of constraints imposed by logical formulae on machine learning processes. *Bottom-up hybrids* invert this relationship, resulting in architectures capable of extracting explicit, structured, often interpretable knowledge by statistical means. Finally, *bi-directional hybrids* feature reciprocal or recurrent feedback of knowledge between subsymbolic and symbolic processes. While for simplicity we speak of these categories in somewhat rigid and mutually exclusive terms, in practice many approaches could be said to exist on a spectrum between these different categories. The SayCan architecture (Ahn et al., 2022), for example, discussed in some detail in Sec. 2.2.5, is decidedly a bi-directional hybrid, yet utilizes an integrative architecture as its primary symbolic component. Similarly, logical neural networks (Riegel et al., 2020), discussed in the following section, utilize representations that, despite being completely neurally realized, map readily to the symbolic systems they seek to emulate. Nevertheless, we believe these categories capture useful functional and structural similarities across otherwise disparate approaches and thus can serve as a means of facilitating comparison among them.

2.2.1 Integrative Approaches

Integrative approaches are monolithic frameworks whereby subsymbolic learners are engineered to perform feats of high-level abstract reasoning, often in the form of symbolic or logical reasoning, by means of carefully designed inductive biases. Overwhelmingly, DNNs and their variants are the subsymbolic learners of choice given

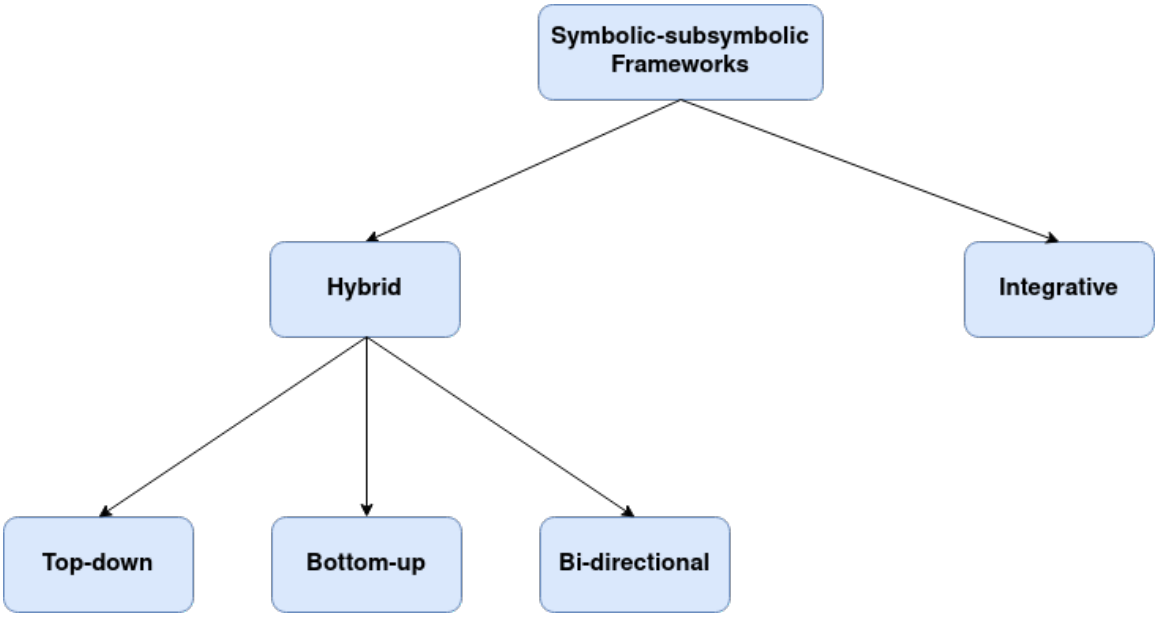


Figure 2.1: A taxonomy of combined subsymbolic and symbolic approaches used across robotics. Central to this taxonomy is the distinction between approaches that are *integrative* – that instantiate and manipulate symbols via subsymbolic processes – and *hybrid* – that contain distinct symbolic and subsymbolic components. Given the separability of knowledge types in hybrid systems, we can further characterize them based on whether symbolic and subsymbolic knowledge is combined in a *top-down*, *bottom-up*, or *bi-directional* manner.

their flexibility. As a consequence, abstract knowledge in these systems is often *distributed* amongst many neurons, rather than *localized* to particular entities or symbols (Garcez and Lamb, 2020), making it a challenge to tease apart the abstractions from the statistical processes that gave rise to them. However, there has been work done to *disentagle* the former from the latter giving rise to techniques for psychological modeling (Page, 2000), or more recently to enable first-order logic (FOL) reasoning in logical neural networks (Riegel et al., 2020) where 1-1 correspondences exist between network structures and the symbols and operations they represent. Indeed, merging FOL and DNNs has been an area of active research in recent times, though typically these approaches employ entangled, continuous vector-based representations of logical theories (Serafini and Garcez, 2016; Payani and Fekri, 2019; Rocktäschel

and Riedel, 2017). The benefit of such approaches is that they unify learning and reasoning, enabling these systems not only to evaluate logical expressions on real-valued, uncertain, or even incomplete data, but in the case of logic tensor networks (Serafini and Garcez, 2016) for example, learn new object relations over which to reason.

Graph Neural Networks. The ability to represent relationships is a desirable feature of traditional symbolic systems that many integrative approaches seek to emulate. Graph Neural Networks (GNNs), for example, are a type of NeSy architecture that extends convolutional neural networks to learn and reason about graph structures (Scarselli et al., 2008; Zhou et al., 2020). This is valuable because graphs are powerful and interpretable abstractions that can represent relational or dependent structures that naturally arise in many domains, including social networks, knowledge graphs, first-order logic, text and image classification. The central intuition behind GNNs is to use the topology of the input graphs to guide the learning of the network's embedding such that closeness or connectedness in graph space results in closeness in the GNN's embedding space. This is accomplished via the process of *neighborhood aggregation*, which updates the latent feature of each node as a function of the node's neighbors in the graph. In practice what this means is that expert or domain knowledge can be provided in a manner not unlike traditional symbolic systems, and it will be reflected at the subsymbolic level of the network. As a consequence, these systems have enabled many feats of abstract reasoning, including causal discovery and inference (Goudet et al., 2017; Yu et al., 2019), spatial reasoning (Giuliari et al., 2022), and scene understanding (Qiu et al., 2021; Zareian et al., 2020). Perhaps most notable is their (indirect) use in natural language understanding and generation by way of large language models trained using transformer architectures (Vaswani et al., 2017; Devlin et al., 2018). While transformers were developed independently of GNNs, the transformer's self-attention mechanism, which learns word representations as functions of other words in the same sentence, lends itself readily

to reinterpretation in terms of the neighborhood aggregation of GNNs. As a result, transformers have been argued to be a specials case of GNNs (cf., Veličković et al., 2017) and many methods for explicitly combining them have been proposed (e.g., Veličković et al., 2017; Nguyen et al., 2022; Yang et al., 2021a).

Reinforcement learning. In the context of reinforcement learning (RL), integrative approaches, such as hierarchical RL (HRL) and model-based RL (MRL), have become increasingly popular as a means for learning reusable, generalizable abstractions that can mitigate the high data costs associated with these approaches.

HRL and MRL both use abstractions to make RL more efficient and effective. In HRL, the concept of temporal abstraction is used to incorporate high-level actions, or options, into the policy learning process. These options are composed of multiple low-level actions and are used to complete sub-components of the overall RL task. This allows the RL agent to learn a hierarchy of policies, with each level of the hierarchy responsible for a different level of abstraction in the task. While many of the foundational HRL approaches, such as the options (Sutton et al., 1999) and MaxQ-Q (Dietterich, 2000) frameworks utilize explicit, pre-specified, and decidedly non-integrative approaches (see Sec. 2.2.2), recent approaches, such as options-critic (Bacon et al., 2017), HAC (Levy et al., 2019), and HIRO (Nachum et al., 2018), have leveraged the power of deep learning to learn these abstractions directly. While these differ in their specific implementational details, they all share the common goal of imposing a hierarchical structure on an agent's policy in order to improve the flexibility and generality of these policies. However, like many integrative approaches, the abstractions learned, namely the options, are subsymbolic as it can be difficult or impossible to know what exactly these options actually do or represent.

In contrast, MRL techniques learn an abstraction at the level of the of environmental dynamics rather than task-level abstractions such as in HRL. In some sense the former is more easily generalizable than the latter as such an abstraction (that

is the transition function T corresponding to an agent's Markov decision Process, or MDP) model's the effects an agent's actions have on the environment, which are not contingent on any particular task. While symbolic approaches to model learning, such as dynamic programming (Chitnis et al., 2021; Rodrigues et al., 2011; Zhuo et al., 2010) are often used in MRL, subsymbolic, deep learning approaches are becoming increasingly popular (see Plaat et al., 2020 for a review). This is because these methods are well-suited for learning complex, nonlinear models, which dynamics models often are, from raw data. The tradeoff is that subsymbolic models are often difficult or impossible to interpret in a human-readable form, whereas symbolic models can be more easily understood and analyzed.

2.2.2 Hybrid Approaches

In contrast to integrative systems, which utilize primarily subsymbolic methods such as DNNs to instantiate and process abstract knowledge, hybrid systems combine two or more problem-solving techniques to address a particular problem (Bader and Hitzler, 2005). This allows hybrid systems to leverage the strengths of individual approaches and overcome their respective weaknesses. In practice, this often means different components of the hybrid are responsible for manipulating knowledge at different levels of abstraction. As a consequence of the disentangled nature of such architectures, asymmetries in the way these components are combined are more readily apparent. More specifically we identify three classes of hybrid: *top-down*, *bottom-up*, and *bi-directional*.

2.2.3 Top-down Hybrid Systems

Top-down hybrid systems combine high-level, abstract, and symbolic representations of a problem with lower-level, more detailed representations and processes. Often this means the use of symbolic knowledge in the form of logical rules or symbolic

structures acting as constraints or guides on the behavior of low-level subsymbolic processes. This architectural organization typically confers two main benefits. First, symbolically defined constraints provide an interface by which users can encode expert knowledge into an agent's learning or behavior. This is an important capability in fields like human-robot collaboration (HRC), where the ability of a robot to integrate commands issued by a user is critical for the success of the collaboration. Indeed, as HRC is the domain of focus of this dissertation, many of the approaches described in subsequent chapters are top-down hybrids. In Chapter 5, for example, we show how a learned model of context – here a graphical representation of a collaborative task – can be leveraged to significantly improve the interpretation of natural language commands. This is accomplished by treating the model as a constraint on the possible interpretations of even highly semantically ambiguous commands. In Chapter 7 we build on this idea by grounding natural language commands in an HRC context to symbolic rules. These rules act as mutable constraints on the robot's policy, enabling a user to modify the robot's policy on the fly.

This latter work draws upon a well-studied top-down hybrid, a variant of Markov Decision Processes known as constrained Markov Decision Processes (CMDPs; Altman, 1999). CMDPs extend the typical MDP formalization by adding the notion of constraints, which act as auxiliary reward functions that penalize the agent for taking actions that violate these constraints. While in the original formulation these constraints were represented numerically, other approaches have enabled constraints to be specified via linear temporal logic (LTL) and other formal languages (see Camacho et al., 2019 for a review). This not only renders such constraints more interpretable but enables a user to issue non-Markovian rewards (Camacho et al., 2017), which can be valuable in a sequential decision-making setting. LTL and other similar languages have also been employed to perform action shielding (Wachi and Sui, 2020a; Alshiekh et al., 2018a). Shielding approaches differ from previously mentioned approaches by

imposing constraints, not on the reward function, but on the actions considered and enacted by the agent. This has the advantage of preventing agents from taking any potentially unsafe actions, rather than merely penalizing them.

The second benefit of top-down hybrids is that symbolic knowledge can speed up low-level processes such as learning or search. In Chapter 4, for example, we demonstrate how tool-based robot skills can be learned quickly by tailoring skill learning based on a taxonomic tool affordance model. Such symbolically mediated speedups are particularly evident in Task and Motion Planning (TAMP; Garrett et al., 2021a), a set of methods for solving the problem of goal-directed motion planning. The main intuition behind TAMP is to first produce a high-level discrete action plan for the particular task, using the generated sequence as action skeletons to guide or constrain optimization or sampling in the low-level configuration space of the robot. While many high-level planning approaches have been suggested ((Garrett et al., 2018, 2020; Pack and Lozano-Pérez, 2011; Gravot et al., 2005) cite), overwhelmingly this planning is performed using symbolic manipulation, often variants of classic GOFAI symbolic planners.

A limitation of many of the approaches described above (and one that appears across many hybrid approaches employing explicit symbolic systems) are their reliance on hand-coded symbols. That is, CMDP and TAMP approaches and their variants typically assume the existence of task-relevant atoms or predicates grounded in the low-level state space of interest. While this may not be a reasonable assumption in more constrained and predictable contexts such as those often found in HRC, it does limit the general applicability and flexibility of these approaches. Some recent approaches (Kumar et al., 2022; Silver et al., 2022) have sought to remedy this issue in the context of TAMP by enabling an agent to learn TAMP predicates from demonstrations. However, such systems, contain a bottom-up component by virtue of the type of learning required, and thus can no longer be considered pure top-down

hybrids.

2.2.4 Bottom-up Hybrid Systems

In contrast to top-down hybrids, bottom-up hybrids are extractive in the sense that the low-level subsymbolic processes produce high-level symbolic representations that are manipulated by some distinct symbol-manipulation system. One benefit of these architectures is that, much like bottom-up hybrids, top-down-hybrids can produce a kind of interface between robots and humans. The difference is directional; the structured representations produced by such systems are transparent and readily interpretable, facilitating a transfer of knowledge from the robot to the user. Outside of explainability, these sorts of representation schemes produce tangible benefits to the reasoning capabilities of robots; abstracted knowledge not only tends to generalize better but can be leveraged to construct other abstractions to enable new feats of reasoning.

Graphical models. Both of these benefits are evident in approaches for structure learning for graphical models (see Drton and Maathuis, 2017 for a review) such as Bayesian network learning (Eaton and Murphy, 2012; De Campos et al., 2009; Tsamardinos et al., 2006) and causal discovery (Spirtes et al., 2000a; Heckerman et al., 1997; Hoyer et al., 2008). Bayesian networks are useful because they create a factorized representation of joint probability distributions in the form of a directed acyclic graph (DAG). This DAG not only encodes interpretable high-level structural information about the variables in the distribution (their conditional dependencies) but also enables more efficient and computationally tractable inference. Causal networks (Pearl, 2018a) are a special case of Bayesian networks where directed edges encode causal relationships, rather than statistical associations. While these networks lose the ability to represent simple associations, they support more advanced types of reasoning capabilities such as interventional and counterfactual reasoning.

These are especially important capabilities in a robotics context, where actions have real-world effects and thus need to be considered accurately and carefully.

Causal reasoning is a cornerstone of another important robot capability, namely the ability to learn and reason about object affordances (Gibson, 1979). As a consequence, learned graphical representations of causal relations are often employed in affordance representations, including DAGs (Montesano et al., 2007), and-or graphs (Xiong et al., 2016), and aspect transition graphs (Ku et al., 2014; Wong and Grupen, 2016). The trade-off, however, with constructing such powerful representations is in their data-intensive nature. This is true both in the sense of the quantity of data required to specify these graphs, but also in quality as well; non-trivial cause-and-effect relationships cannot be learned from observational data alone, but require interventional data (Pearl, 2000).While robots are well-positioned to collect such data as a consequence of their physical embodiment, collecting such data can be time-consuming and infeasible in some scenarios. To mitigate this challenge, in Chapter 3, we demonstrate a method by which a robot can efficiently construct a causal DAG via self-guided exploration. The robot then uses this causal model to construct tool-affordance models in order to wield familiar and novel tools.

Symbol grounding. In the approaches described in Chapters 6 and 7, we assume the existence of interfaces that can transform simple user speech commands into logical expressions. These logical expressions enable users to refer to particular objects in the environment by associating these objects with particular predicates that constitute these expressions. However, the problem of learning the connection between language or symbols and their real-world referents, the symbol grounding problem (Vogt, 2002), is a challenging one, though many bottom-up hybrid solutions have been proposed. Many of these approaches accomplish this by learning associations between language input and some intermediate representation of the environment, such as spatio-temporal graphs (Alomari et al., 2017), scene graphs (Yi et al., 2022), or

logical formalisms such as relational logic (Wächter et al., 2018; Hayes and Scassellati, 2016). In terms of the latter, LTL is a popular grounding formalism (Gopalan et al., 2018; Berg et al., 2020; Hsiung et al., 2022a) due to the correspondence between LTL expressions and Buchi automata (Büchi, 1990). As a consequence, unlike relational logic, an LTL-grounded user command can readily be transformed into a multi-step, potentially non-Markovian robot plan. The tradeoff with these LTL approaches is that they do not fully solve the symbol grounding problem; even if some approaches can ground commands to LTL expressions that can generalize to previously unseen predicates (Hsiung et al., 2022a), it is still assumed that these predicates representing high-level state features exist. Approaches like (Alomari et al., 2017) in contrast can ground commands more directly to lower-level features of the state-space, but they lack LTL-based approaches' ability to interpret and act on commands requiring multiple steps to complete.

Reinforcement learning. In reinforcement learning, systems that can learn abstractions in a bottom-up way are pervasive. These not only include non-integrative variants of HRL and MRL approaches that learn action and transition function symbolic abstractions, respectively, but also state and policy abstractions as well.

In terms of action abstraction, as Konidaris (2019) observes, many of the approaches rest on the ability to identify sub-goals from which options can then be constructed. This has been done using, for example, graph-theoretic analysis on topological state features (Menache et al., 2002; Mannor et al., 2004; Kazemitabar and Beigy, 2008), or by identifying high-reward or high novelty states (Stolle and Precup, 2002; Şimşek and Barto, 2004). Bottom-up MRL approaches have historically focused on learning symbolic relational transition models (Chitnis et al., 2021; Rodrigues et al., 2011; Zhuo et al., 2010; Lyu et al., 2019). Such models represent transition functions in terms of STRIPS-style action rules (Fikes and Nilsson, 1971) comprised of lifted relational predicates, producing a generalized description of the

actions available to the agent. The trade-off between the option learning approaches on the one hand and the MRL approaches on the other is a trade-off between flexibility and generality. Assuming a constant environment, symbolic STRIPS-style action rules can readily generalize across tasks as such representations do not contain any task-dependent features. However, they assume the existence of symbolic state and action features, which may not always be guaranteed in complex real-world scenarios. Option learning approaches do not make assumptions, and so are more widely applicable. However, there are no guarantees that options will be useful for completing new tasks, as they are learned in an intratask context.

Techniques for policy abstraction attempt to learn a policy representation in such a way that it can more readily be applied across task instances. In relational reinforcement (RRL) policies are learned as sequences of lifted relational actions that can more generalize to different task instances (Kersting et al., 2004; Driessens and Ramon, 2003). More recently there has been a concerted effort to engineer deep reinforcement learning architectures capable of representing their policies in symbolic terms so as to make them more explainable and transferable. These include methods for transforming DRL policies into decision trees (Gupta et al., 2015; Roth et al., 2019; Coppens et al., 2019), or algebraic expressions (Landajuela et al., 2021). RRL methods are powerful because their ability to generalize come 'for free.' That is, by learning policies in terms of relational predicates, transferring them to a new task instance is tantamount to finding a new grounding for the predicates that satisfy this new context. However, as with the symbolic MRL approaches referenced above, this assumes that states and actions can adequately be represented in terms of these predicates, which may not always be feasible in practice. Explainable DRL methods, in contrast are much more flexible in the sense that they are not limited to learning in symbolic state and action spaces, since the symbolic policies are extracted from subsymbolic policies. However, producing such representations requires large amounts of

data and are not guaranteed to model the underlying policy completely accurately.

Approaches to state abstraction typically attempt to find a simplified state representation such that solutions learned in the new space readily transfer to the original space. This has been accomplished via feature selection methods that select only the most relevant features from which to construct a condensed state representation (Kroon and Whiteson, 2009; Johns et al., 2010; Wookey and Konidaris, 2015). Other approaches perform what Konidaris (2019) deems *representation discovery*, which produces a kind of isomorphic mapping from the original state space to a novel, but more condensed alternative (Mahadevan et al., 2006; Scholz et al., 2014; Jonschkowski and Brock, 2015). Feature selection methods are typically fast and efficient, but always risk the possibility of removing potentially valuable state features from consideration. Representation discovery approaches, on the other hand, have the potential to uncover new representations that may not have been explicitly specified by the designer. However, these approaches may require more data and computation to learn and may be more difficult to interpret and understand than the original state space.

2.2.5 Bi-directional Hybrid Systems

Bi-directional hybrid systems are those that combine both top-down and bottom-up processing, allowing the system to operate at multiple levels of abstraction simultaneously or interact in a cyclical manner. As described in the proceeding sections, bottom-up and top-down architectures endow a robot with complementary capabilities. That is, bottom-up approaches produce transparent, reusable abstractions from subsymbolic processes such as those responsible for perception. Top-down approaches integrate these abstractions or abstractions produced via their own reasoning into the subsymbolic processes that enable the robot to interact and navigate the world. As a consequence of the flexibility and adaptability entailed by combining these approaches, hybrid architectures most often manifest as general-purpose architectures

or architectures designed with the intention of being deployed in the real world.

General-purpose architectures. Given that cognitive architectures attempt to model the general capabilities of the human mind, they overwhelming employ a bi-directional approach. While cognitive architectures like SOAR (Laird et al., 1987) and ACT-R (Anderson et al., 1997), can differ significantly in terms of underlying design philosophies, they share fundamental similarities. These techniques can generally be described as modular architectures wherein a central, explicit knowledge base is modified both in a top-down manner via modules that apply high-level, symbolic production rules (though the application of such rules are often dependent on the state of the knowledge base itself) and in a bottom-up manner via extractive perceptual modules. Early iterations of these architectures modules primarily learned via the top-down creation of new production rules, though subsequent iterations would add more sophisticated bottom-up learning capabilities, such as the ability to use language models as a knowledge source in SOAR (Wray et al., 2021).

More recent cognitive architectures like CLARION (cf. Chong et al., 2007) and DIARC (Scheutz et al., 2019) were designed from the outset to employ multiple interconnected knowledge representational schemes and learning approaches. CLARION, for example, is a two-layered architecture where the top layer contains symbolic propositional rules and the bottom layer contains subsymbolic procedural knowledge acquired via Q-learning. Under this scheme, subsymbolic knowledge may ultimately be promoted to a symbolic rule if the former produces high-reward actions. DIARC extends this notion of multiple learning strategies by enabling each component of the architecture to employ its own learning strategy and representational scheme based on the types of information they process. It is this architectural flexibility in addition to a privileging of motor and perceptual components that have made DIARC especially well-suited for robotics applications (Brick et al., 2007; Brick and Scheutz, 2007; Sarathy et al., 2016; Talamadupula et al., 2014), though variants of SOAR (Hanford

et al., 2009), ACT-R (Trafton et al., 2013), and CLARION (Ghayoumi and Pourebadi, 2019) have all been employed on robots as well. Of note as well is DIARC's use of a bespoke probabilistic logical formalism for learning and reasoning about so-called *cognitive affordances* such as social norms (Sarathy et al., 2017). In Chapter 6 we detail our own bi-directional hybrid architecture capable of learning and utilizing norms related to ownership using an analogous probabilistic logic approach.

In terms of non-cognitive generalist hybrid architectures, SayCan (Ahn et al., 2022) offers a notable recent example. Of particular interest is its use of the PaLM (Chowdhery et al., 2022) LLM for top-down symbolic planning and reasoning, rather than a symbol-manipulation approach employed by most cognitive architectures. This is accomplished in part using a modified version of chain-of-thought (COT) prompting (Wei et al., 2022), a training technique that shapes LLMs into producing answers to queries that include intermediate reasoning steps leading to the answer. Another important component is a series of low-level policies, or skills, grounded in a bottom-up manner to natural language commands (e.g., "open the drawer" or "place the can down upright"). By training the LLM to produce such reasoning steps only in terms of these commands, the model can flexibly produce a series of behavioral steps the robot can actually enact in response to high-level natural language human commands.

The advantage of SayCan over most cognitive architectures is in its flexibility and extensibility; extending the capabilities of the system is tantamount to training a new low-level skill and making its linguistic grounding available to the LLM. In contrast, though more recent cognitive architectures like DIARC were designed with extensibility in mind, adding capabilities is typically much more involved, usually requiring additions like hand-coded production rules for each new task or skill. On the other hand, as Wei et al. (2022) admit, there are no guarantees that the plans produced by COT prompting will be accurate or coherent. This is somewhat mitigated by SayCan's affordance model that scores candidate plan steps based on their feasibility

in the current context. However, this still falls short of the reliability of precision of plans generated via the deterministic and transparent production rules of most cognitive architectures. Only time will tell if the flexibility/reliability trade-off is a worthwhile one for general-purpose robots deployed in the real world.

Self-driving architectures. Architectures responsible for controlling self-driving vehicles are almost universally bi-directional architectures (see (Hsiung et al., 2022b) for a review). This is because despite differences in the particulars of each architecture, at a high level they have converged on a single solution: construct a detailed internal representation of the environment from perception and use this model to plan a path to a user-defined destination. This naturally lends itself to a combined bottom-up and top-down approach, as the former excel at building useful abstractions from low-level data and the latter excel at deterministic and proveably correct manipulations of such abstractions.

In order to construct a robust and detailed internal world model multiple perception systems are typically employed including RGB cameras, LIDAR, and RADAR (e.g., Xue et al., 2017) as well as GPS and car-odometry (e.g Aeberhard et al., 2015). From these varied perceptual inputs useful features are estimated including. but not limited to the pose of the vehicle (Brubaker et al., 2015; Wolcott and Eustice, 2017; Hata and Wolf, 2015); a map of the environment, usually represented either topologically (Cummins and Newman, 2008; Forechi et al., 2018) or using metric representations (Mutz et al., 2016; Schaefer et al., 2018); the pose of moving objects in the immediate environment (Petrovskaya et al., 2012; He et al., 2016; Mutz et al., 2017).

Planning in an autonomous vehicle leverages this internal model, typically across multiple sub-modules, in order to drive itself safely and efficiently to a desired location. The correctness and reliability of plans are especially crucial in a self-driving context, where a wrong behavioral decision can have truly disastrous consequences,

which is why symbolic approaches are often employed. For example, variants on classic graph-based search algorithms like Dijkstra (Dijkstra, 2022) and A-star (Hart et al., 1968) are used in route planning (Goldberg and Harrelson, 2005; Gutman, 2004) and for generating candidate paths within those routes (Arnay et al., 2016; Ziegler et al., 2008). Behavior selector subsystems, which arbitrate between candidate paths and decide lower-level behavioral strategies also employ symbolic and graphical approaches such as finite state machines (Montemerlo et al., 2008; Jo et al., 2015), ontology-based knowledge bases (Zhao et al., 2015), MDPs (Brechtel et al., 2011), and partially-observable MDPs (POMDPs; Galceran et al., 2017; Ulbrich and Maurer, 2013).

Today, self-driving vehicles are arguably the most sophisticated robots a person can expect to encounter in everyday life. Despite this, based on the Society of Automotive Engineers' 5 level taxonomy of autonomy (SAE, 2018) where a level 5 rating represents a car requiring no human intervention in any circumstance, the vast majority of cars achieve only a level 3 rank (Hsiung et al., 2022b). The challenges that remain are both perceptual and planning related. Perception systems struggle with reliably detecting and interpreting objects or events that are occluded or partially visible, especially with changing lighting conditions or adverse weather. On the planning side, unsolved ethical dilemmas exist such as how behavior selector subsystems should adequately arbitrate between paths that lead to potentially life-threatening outcomes.

2.3 Summary and Discussion

Thus far we have discussed the functional and representational differences between subsymbolic and symbolic knowledge, provided an overview and taxonomy of architectures for combining these forms of knowledge, and discussed how these architectures

have been employed across robotics. In discussing the taxonomy of approaches, we identified a fundamental dichotomy between approaches that are integrative – that instantiate and manipulate symbols via subsymbolic processes – and hybrid – that contain distinct symbolic and subsymbolic components. In so doing, we are able to unify a wide variety of approaches across robotics under a single unified framework, facilitating comparisons between once unrelated domains of research.

While we consider the broad applicability of this taxonomy to be a strength, it is also something of a limitation; by necessity, its categories must also be broad, limiting the sorts of finer-grained analyses enabled by more specialized taxonomies. This is most evident in our integrative category, which elides the nuances that may exist between these approaches. However, we see our taxonomy as complementing, rather than replacing, specialized NeSy taxonomies such as those presented in Kautz (2022) and Bader and Hitzler (2005).

On the other hand, the broadness of this category enables trends and thus fruitful areas of future research to be readily identified. For example, from our taxonomy, transformer architectures emerge as an extremely popular choice across both integrative (e.g., Devlin et al., 2018; Reed et al., 2022; Lee et al., 2022; Storks et al., 2021) and hybrid approaches (e.g., Ahn et al., 2022; Wray et al., 2021; Singh et al., 2022). It is clear that the language capabilities of these sorts of models have achieved a level of sophistication such that it is difficult to justify their lack of inclusion in any language-based robotics or AI system going forward. Similarly revealed by this taxonomy is the surprising re-emergence of STRIPS-style rules, particularly PDDL-based ones (e.g., Lyu et al., 2019; Konidaris et al., 2018; Chitnis et al., 2021; Verma et al., 2021; Silver et al., 2022) as powerful tools for enabling greater levels of explainability and knowledge transfer. Despite peaking in popularity in the the 1970s and 80s, this trend suggests that STRIPS is not a relic of a bygone era but perhaps a foundationally important technique to robotics.

Nevertheless, an obvious question remains: do integrative and hybrid approaches, and the sub-categories therein, all have a place in robotics, or should one set of approaches be the focus of the field? We argue that there is no single ideal category, but rather each offers techniques that may be useful depending on the application. For example, robotics applications where safety is absolutely paramount could benefit from the use of top-down approaches such as action shielding and CMDPs (refer to Sec. 2.2.3). Likewise, if flexible and powerful language capabilities are desirable, integrative LLM-based approaches may be ideal. However, in terms of which of these broad categories is most likely to produce intelligent and practical real-world robots, we believe the evidence overwhelmingly points to hybrid systems. Recent history has already shown this to be the case; vehicles with autonomous driving capabilities, arguably the most sophisticated mass-produced robotics systems, overwhelmingly employ bi-directional hybrid architectures (refer to Sec. 2.2.5 and see Hsiung et al., 2022b for a review). Similarly, as Marcus (2020) argues, many of the most successful commercial AI systems, notably Google Search, owe their success to use of hybrid architectures.

As a result, the novel work presented in subsequent chapters primarily employs hybrid approaches in order to improve various aspects of HRC. We begin our discussion of our work in Chapter 3, where we present our hybrid approach for learning and utilizing causal relations for tool affordance reasoning.

Chapter 3

Bottom-up Hybrids: How Can Robots Learn Useful Abstractions?[1]

In this chapter, we explore the bottom-up learning of symbolic abstractions for improving a robot's ability to reason about and manipulate its environment. Specifically, we present a method by which a robot can learn a structural causal model of simple tool use via self-supervised experimentation. Using these learned causal relations, the robot is able to construct a tool-affordance model for each available tool. Our results with a physical robot demonstrate that these abstractions can be used to complete tool-selection and goal-directed tool-use tasks with familiar and novel tools using minimal training data.

[1]Portions of this chapter were originally published as: J. Brawer, M. Qin, and B. Scassellati. A causal approach to tool affordance learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8394-8399. IEEE, 2020. (Brawer et al., 2020)

3.1 Introduction

In the fable *The Crow and the Pitcher*, the ancient Greek poet Aesop tells the story of a thirsty crow who, by dropping stones into a jug of water, raises the water level high enough to drink from. Recently, the scientific community corroborated this remarkable feat of physics-based instrumental problem-solving in New Caledonian Crows (NCC; *Corvus moneduloides*; Jelbert et al., 2014). Contrary to the flexible reasoning capabilities of crows and other higher species, robot learning and reasoning remains starkly limited despite substantive advancements in statistical machine learning techniques. Additionally, the learning that is acquired through these systems remains largely opaque (Marcus, 2018), which not only makes the systems difficult to debug but poses serious risks in robotics settings where unforeseen behaviors can cause physical harm.

Ideally, robots should be able to acquire knowledge and skills in a way that is both transparent and transferable across contexts without compromising the incredible advancements made by the machine learning community. To that end, we have developed a system whereby a robot can learn and exploit a graphical causal model to solve a physical reasoning task. Our approach has a robot learning a structural causal model (SCM; refer to Sec. 3.2.1 for a formal definition) through a mix of observation and physical exploration. The SCM is used to perform causal inference, which is completed by a group of neural networks that are dynamically constructed and trained as a function of the learned structure of the SCM and the goals of the current task. As a result, our system represents the robot's knowledge in an explicit and explainable way via the directed acyclic graph (DAG) entailed by the SCM. It also leverages the powerful pattern recognition capabilities of machine learning techniques via the use of neural networks.

In this chapter, we provide a proof-of-concept demonstration our method (see Figure **??** for an overview) on a humanoid robot that must build a model of the

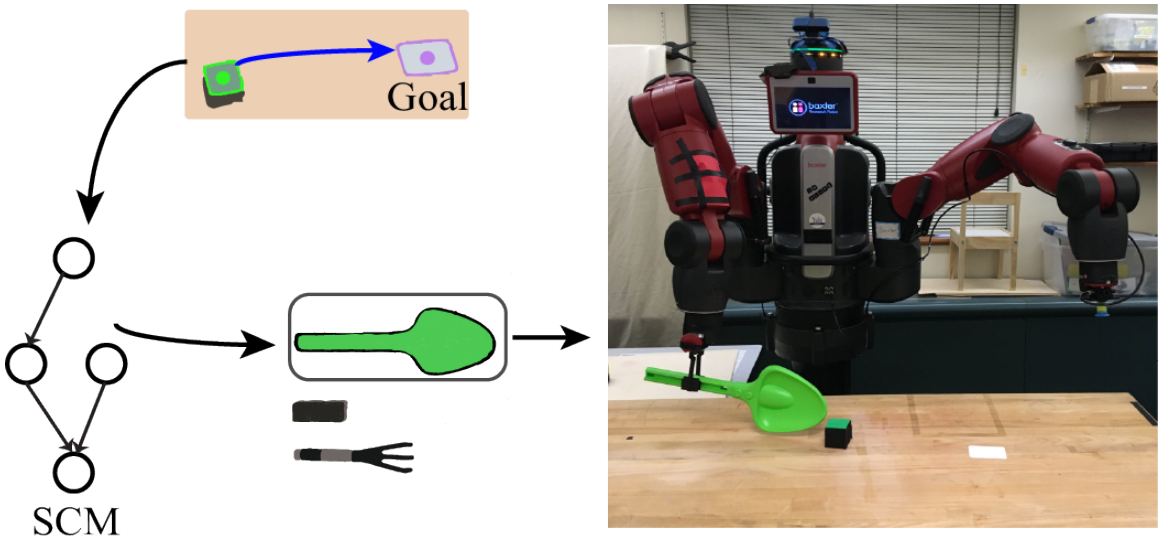


Figure 3.1: Causal inference can be used to perform tool selection. A Baxter collaborative robot queries a learned causal model of tool-assisted manipulation using perceptual information from its workspace. Information from the graph is used to select the most appropriate tool for completing its goal.

cause-and-effect relationships underlying tool-assisted manipulation. We then use this model to both solve goal-directed manipulation tasks and quickly learn the affordances of novel tools given a kinematic model of each tool.

We demonstrate that our system is capable of selecting the appropriate tool and action to move an arbitrarily placed block into goal regions, as well as leveraging prior learning to bootstrap learning of new tools.

In sum, our contributions are the following:

- A method for a robot to construct a transparent model of causation via a mix of observation and experimental learning.
- A method for performing forward and backward causal inference using a series of dynamically constructed neural networks.
- A method for a humanoid robot using learned causal models to quickly learn and utilize tool affordances of novel tools.

3.1.1 Causality in Machine Learning and Robotics

Many researchers have attempted to formalize causal relations over the past century. Here we focus on Pearl's SCM framework to formalize causal relations using directed acyclic graphs (DAGs) that define structural equations between causal variables (Pearl, 2000). Pearl argues that SCMs accommodate sophisticated forms of reasoning, including interventional (e.g., "What if I had done X?") and counterfactual (e.g., "What if I had acted differently?"; Pearl, 2018b). Counterfactual reasoning has been integrated into reinforcement learning (RL) systems; for example, it has been shown to enable the acquisition of effective decentralized multi-agent policies in a credit assignment task (Foerster et al., 2018), and aid in the evaluation of high-risk healthcare policies (Oberst and Sontag, 2019).

SCM-based reasoning has been employed in a robot by Angelov et al. (2019) which used learned SCMs to counterfactually reason about user preferences in their demonstrations of a motion task. Non-SCM-based approaches to causal reasoning include Xiong et al. (2016), which integrated spatial, temporal, and causal and-or graphs learned from user demonstrations to enable a robot to fold clothing. Nevertheless, causal learning and reasoning for robots remain relatively unexplored despite the active developments from the machine learning and artificial intelligence communities.

3.1.2 Affordance Learning

While Bayesian networks have been a popular method for representing affordances (see Andries et al., 2018 for a recent example), an explicitly causal framework is rarely employed despite the centrality of cause and effect within many affordance paradigms. As frameworks like SCMs can tease apart causal, and not merely correlational, properties in the environment, a robot equipped with such a framework is better poised to model the effects of its actions on the world, and thus to acquire and utilize affordances. We demonstrate this capability by showing that our system

can effectively complete goal-directed tool and action-selection tasks using acquired affordance knowledge.

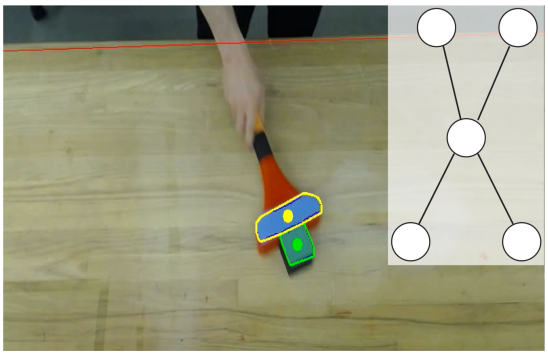
3.2 Methods

3.2.1 Problem Statement

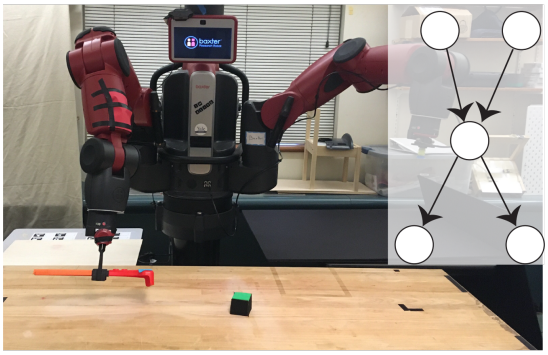
Our goal is to have a robot model simple tool use behaviors using SCMs, and then subsequently use this model to quickly ascertain the affordances of novel tools. We begin by giving a brief overview of the SCM formalism (refer to Pearl, 2000 for more technical details).

The problem of learning an SCM is twofold: the structure of the graph which links variables to other variables – or causes to effects– must be learned, as well as the functional mechanism that specifies these relationships. Formally, an SCM $\mathcal{M}$ is a tuple $\{\mathcal{U}, \mathcal{V}, \mathcal{F}\}$ where $\mathcal{U}$ is a set of unobserved background features or "exogenous" variables, $\mathcal{V}$ is a set of observed features or "endogenous" variables, and $\mathcal{F}$ is a set of functions that assigns values to variables in $\mathcal{V}$ based on other variables in $\mathcal{U}$ and $\mathcal{V}$. Under the assumption that the causal structure is acyclic, there is a corresponding DAG $\mathcal{G}$ where the nodes in the graph correspond to variables in $\mathcal{U}$ and $\mathcal{V}$ and the edges to the functions in $\mathcal{F}$.

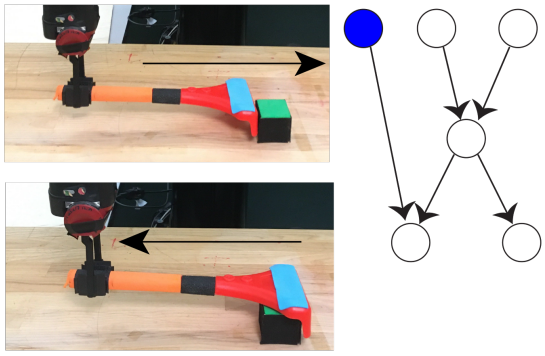
Kocaoglu et al. (2017) demonstrated that feedforward neural networks admit an interpretation as SCMs. We take advantage of this fact by using the causal structure uncovered during learning to guide the construction of a series of neural networks that define the structural equations $\mathcal{F}$ underlying the SCM. The advantage of using such a scheme is that it minimizes the assumptions of the generating distributions of the variables of interest, which in many potential real-world scenarios are likely unknown a priori. Exogenous variables $\mathcal{U}$ are represented by the hidden nodes in the neural network.



(a)



(b)



(c)

Figure 3.2: The robot learns a causal model in three phases. During the observation phase (a) the robot learns a skeleton of the causal graph by observing demonstrations performed by a human. During the validation phase (b) the robot attempts to orient the edges of the graph via self-supervised experimentation. Finally, during the augmentation phase (c), the robot introduces a new node (blue) and attempts to incorporate it into its graph via further experimentation.

3.2.2 Our Approach

Causal Learning

Depicted in Figure 3.2 is our three-phase causal learning approach: the **observational phase** to learn an unoriented skeleton of the causal graph, the **validation phase** to orient the edges of the graph, and the **augmentation phase** to augment the graph with new nodes via experimentation. The goal of the observational phase is to bootstrap learning of causal structure by taking advantage of the fact that it can be inferred (usually within an equivalence class of DAGs under reasonable assumptions of causation, see Eberhardt, 2017 for an overview of causal discovery) from passively collected, observational data using standard structural learning algorithms. This helps minimize the amount of interventional data required to fully specify the graph, which may be beneficial, as often collecting this sort of data is difficult to collect for reasons of practicality. The result of this phase is an undirected graph with edges drawn between causally dependent nodes, though the direction of causation may not be known.

Subsequently, the robot enters the self-supervised validation phase, wherein it attempts to orient the graph by collecting interventional data. An edge between nodes V_1 and V_2 is oriented $V_1 \rightarrow V_2$ if $P(V_2|do(V_1 = x)) \neq P(V_2|do(V_1 = y))$ for some interventions x and y, and vice-versa. As interventional samples can be difficult or costly to obtain, it is desirable to be able to preferentially select nodes to intervene on such that the total number of experiments is minimized. To that end we take an active learning approach adapted from Hauser and Bühlmann (2014), which simply intervenes on nodes in the graph produced during the observational phase starting with the nodes with the highest degree, ensuring that the most informative interventions are carried out first.

Finally, in the augmentation phase, the robot attempts to incorporate a new node

V_3 into its causal model. Unlike in the validation phase, the goal of the augmentation phase is not orienting edges that already exist, but rather adding new oriented edges where none had existed previously. An edge between a new node V_3 and an extant node V_1 is added to the model if it is determined that intervening on V_3 affects the value of V_1 using the method described above.

In order to minimize the number of edges to be tested, and thus the chances of falsely identifying causal relationships, we developed a few heuristics for selectively testing nodes. First, nodes are tested in topologically sorted order. This ensures that the causal antecedents that have thus far been identified are tested first. In addition, we make the following assumption: if an edge $V_1 \rightarrow V_2$ is found for some nodes V_1, V_2, then we do not test any descendants of V_2. While this has the potential to produce an incomplete causal graph, it avoids the possibility of mistaking indirect causation from $V_1 \rightarrow V_2 \rightarrow V_3$ for some descendant V_3 of V_2, for the direct causal connection $V_1 \rightarrow V_3$.

Causal Inference

Once the causal structure is in place, the functional mechanisms underlying the SCM can be learned, allowing the robot to perform causal inference. As we wish to minimize our assumptions about these underlying mechanisms, we use neural networks to estimate these functions, using the structure of the graph to guide the structure of the neural network architectures. However, while this allows for reasoning from causes to effects, it is not immediately clear how other forms of causal inference, e.g., diagnostic or "abductive" reasoning from effects to causes, can be performed. Similarly, in many real-world scenarios, it is often the case that observations have been made for only a subset of variables in the SCM, and it may be desirable to estimate the unknown values using known information.

In order to support these capabilities, the final set of neural network architec-

Algorithm 1 Recursive procedure for causal inference.

```
1: procedure AnswerQueries(Q, G)
2:     if Q is empty then return
3:     else
4:         (q, score) = max(causalScores(G, Q))
5:         if score == 1.0 then
6:             Input = Pa(q)
7:         else
8:             Input = V_observed ∪ V_colliders
9:         q̂ = f(Input)
10:        Append q̂ to G.obs
11:        AnswerQueries(Q \ q, G)
```

tures is dependent on not only the causal graph structure G, but the observed and unobserved variables as well. We treat these unobserved values as a set of "queries" Q, the values of which we would like to estimate. Each node $q \in Q$ is ranked with a "causal score" that is used to ultimately guide the order of inference, as well the construction of the neural networks via a recursive algorithm (refer to Algorithm 1). A causal score is a value between 0 and 1, computed as the ratio of the number of q adjacent nodes with observed values to the total number of q adjacent nodes. If the values of all of q's parents are observed, it is automatically assigned a score of 1, as its value can be estimated by standard means. Once the scores are computed, the value of the node with the highest score is inferred, and it is treated as an observed variable, and so can be used to infer the values of other nodes. This process of scoring and inference is repeated until all nodes have been estimated.

Inferring the q values was accomplished using feedforward neural networks, though the structure of these networks may not match the forward structure of the corresponding causal graph in cases where abductive inference is required. We use two heuristics to construct each network f; 1) if all of q's parents are observed, then $q = f(Pa(q))$, where $Pa(X)$ denotes the parent nodes of node X; 2) otherwise, $q = f(V_{observed} \cup V_{collider})$, where $V_{observed}$ is the subset of nodes adjacent to q that

have been observed, and $V_{collider}$ is the possibly empty subset of nodes that belong to a collider subgraph with q, as these dependencies may carry useful information about q's state. Thus the size of the input and output layers of any neural network was equal to the number of nodes in $V_{observed} \cup V_{collider}$ and the number of nodes in q (i.e. 1), respectively. In practice, we found that a single hidden layer consisting of 10 nodes worked well across all networks.

Affordance Learning

Andries et al. (2018) identify equivalences between affordances based on how similar their effects are on an object when acted upon. We employ a similar approach to learn and represent tool affordances. For some tool t and set of possible actions A, we represent its affordances as a vector $\mathbf{a^t} \in \mathbb{R}^{|A|}$, where each element a_i^t of the vector corresponds to a measure of how well the effects of a tool action align with predictions (here we use the coefficient of determination, r^2). Using a learned causal model of an exemplary tool, a robot can quickly estimate these vectors from a small number of exploratory actions on an object and use these estimates for tool selection and usage given some goal position for the target object.

3.2.3 Experiments

In this section, we discuss the design and implementation details of the task used to evaluate our model.

The Task

We would like the robot to 1) model the causal process underlying manipulating objects with tools, and 2) leverage the learned model to learn and wield new tools. Here we take as examples pushing and pulling as classes of actions the robot can perform. We assume the robot can observe the random variables corresponding to the

initial position and final position of a manipulated block, p_{init} and p_{final}, respectively, as well as the movement vector d, which is defined as the vector that the tool tip travels starting from a small displacement before or after the block to the terminus of the push action. While $p_{init}, p_{final}, d \in \mathbb{R}^2$, we represent each of their dimensions by their own individual nodes, which we express with a superscript, for example, p^x_{init} and p^y_{init}. For convenience, when we omit the superscript, we are referring to both dimensions simultaneously. Additionally, while $d \in \mathbb{R}^2$, for simplicity, we limit the push vectors the robot can enact to 12 evenly spaced vectors around the unit circle. This requires that for any estimated value of d, the robot must choose one of these 12 vectors with the smallest angular distance to d to enact. The robot can take actions $push, pull \in A$ in directions d, though does not at the outset know the relationship between these two, or any other variables. Consequently, during inference, d and A are estimated using classifier networks, while p_{init} and p_{final} are estimated using regressor networks.

While it is assumed the robot knows how to perform simple pushes and pulls with each tool, it does not know the effects that tool use entails, nor in what scenarios a given tool is appropriate. While the robot only needs to learn the causal relationships among these 7 variables, the number of possible DAGs is super-exponential in the number of variables (Robinson, 1973), for a total of approximately 1.1×10^9 possible DAG structures, making this structural identification problem non-trivial.

In addition, there are a number of details related to the implementation of three learning phases that are specific to this particular task, outlined below.

Observational Phase: The robot observed demonstrations of a block being pushed with a hoe tool (see Figure 3.3) by a human, and tracks p_{init}, p_{final}, and d. This observational data is standardized and passed to a structural learning algorithm in order to get an initial hypothesis of the causal structure. For the purposes of this experiment, we use the PC algorithm, a widely used score-based method for causal

discovery (Spirtes et al., 2000b).

Validation Phase: We allow the robot to directly intervene on p_{intial}, p_{final} and d, and limit it only to pushes with the hoe from the previous phase. Each intervention is treated as a randomized controlled trial, where the intervened variable is forced to take on one of two values. Interventions on p_{init} and p_{final} compare interventional distributions for two prespecified positions, whereas interventions on d compare the distribution induced by taking a random action vs taking no action at all. As we wish to limit the assumptions we make about the underlying distributions of the variables, we use the Kolmogrov-Smirnoff test (Daniel, 1990) to nonparametrically estimate difference between the two interventional distributions. Given that these interventions are used to quickly infer causal relations, and are not themselves rigorous scientific experiments, the $p < .05$ significance convention need not be followed. Here we relax the threshold of significance to $p < .2$, though this may be treated as a hyperparameter that trades-off risk of type I errors for data-efficient estimation.

Augmentation Phase: During the augmentation phase, the robot attempts to add the action type A to its causal model. This proceeds in much the same way as the validation phase, except we allow the robot to perform pulls as well as pushes.

Affordance-based Tool Selection

We would like the robot to choose the best tool, $\hat{t}$, and tool usage $\hat{d}$ given the circumstances of the environment. Using desired movement vector d^*, obtained by querying the causal graph, together with the estimated affordance information $a_x^{t_i}$ for a given tool t_i, the robot can make this selection. This is captured by the heuristic

$$\hat{t}, \hat{d} = \underset{t_i, d_{t_i}}{\arg\min} (2 - a_x^{t_i})(2 - \cos\theta_{d_{t_i} d^*}) \tag{1}$$

Here d_{t_i} is a movement direction the robot is capable of actuating with tool t_i

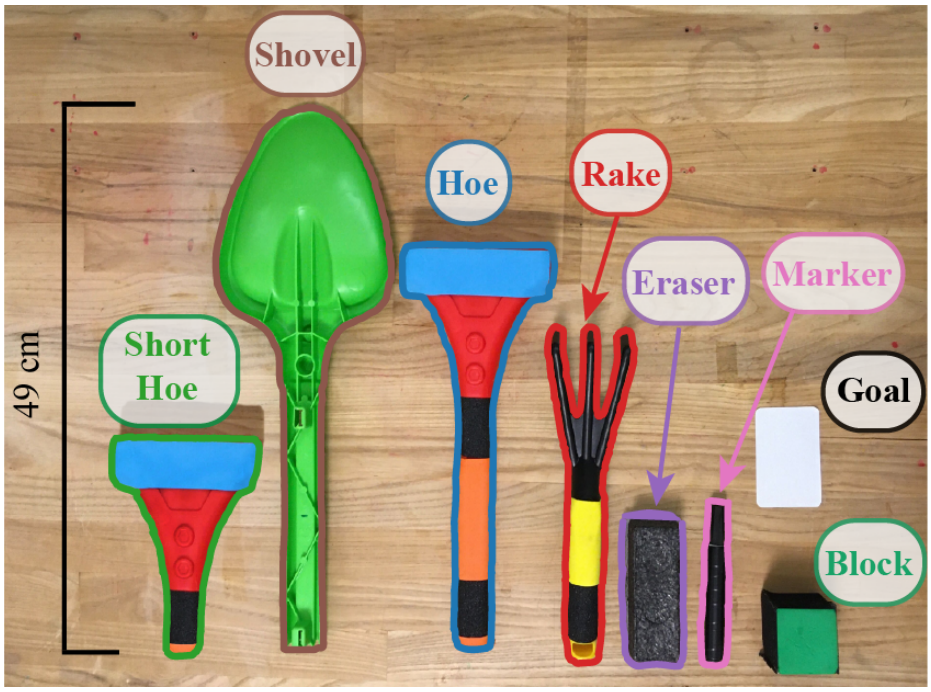


Figure 3.3: The robot had access to 6 tools during our experiments. The tools were used to push and pull a block into a goal region of the workspace. Outlines have been added to color-code each tool and aid in the interpretation of subsequent figures in this chapter.

according to its kinematic model. In essence, this heuristic chooses a tool based on the trade-off between how well it affords a desired action in general, and how well it can enact the action in this specific instance.

The Workspace

Experiments were performed on a Baxter collaborative robot (Figure 3.1)[2]. Our Baxter model was equipped with a claw gripper for manipulating the tools, as well as a suction gripper for picking and placing the block during the self-supervised learning phases. Informal tests suggested our model's end-effector precision was within $\pm 1cm$. Figure 3.3 depicts the contents of the robot's workspace. Actions were performed on a $5cm^3$ wooden block. Initially, the robot had access only to a hoe. Additionally,

[2]Source code can be found at https://github.com/ScazLab/crow

there were 5 morphologically distinct tools used to evaluate the robot's learning. During the evaluation phase, the robot was tasked with pushing the block into a small rectangular goal region measuring approximately, $10cm \times 8.5cm$. The positions of the tool tips, the block, and the goal region were tracked with a head-mounted RGB webcam using a blob detector calibrated to the colors unique to each object. We used the Causal Discovery Toolbox's (Kalainathan and Goudet, 2019) implementation of the PC algorithm, as well as classifier and regressor neural network implementations from Scikit-learn (Pedregosa et al., 2011).

3.2.4 Evaluation

With our experiments, we wished to see how well a robot could use a learned causal model to perform affordance-based reasoning for familiar and unfamiliar tools. In order to do this, we ran three evaluations: 1) Given 8 samples with each of the 5 novel tools, we had the robot choose both the best tool and action to perform given a single fixed goal location, but arbitrary initial positions of the block; 2) given 20 training samples per tool, multiple goal regions, and multiple initial block positions, we observed how close could the robot move the block towards the goal region for each tool; 3) we looked at how accurately the robot could predict action effects as a function of training data samples for a subset of the 6 tools.

3.3 Results

Figure 3.4 depicts the learned causal graph. The structure of this graph models two important aspects of the task: 1) the final position of the block is a function of the initial position of the block and the direction in which it is pushed, and 2) the type of action, push or pull, effects the push direction, suggesting the robot has successfully grounded these abstract actions to a concrete sensorimotor effect. The observation

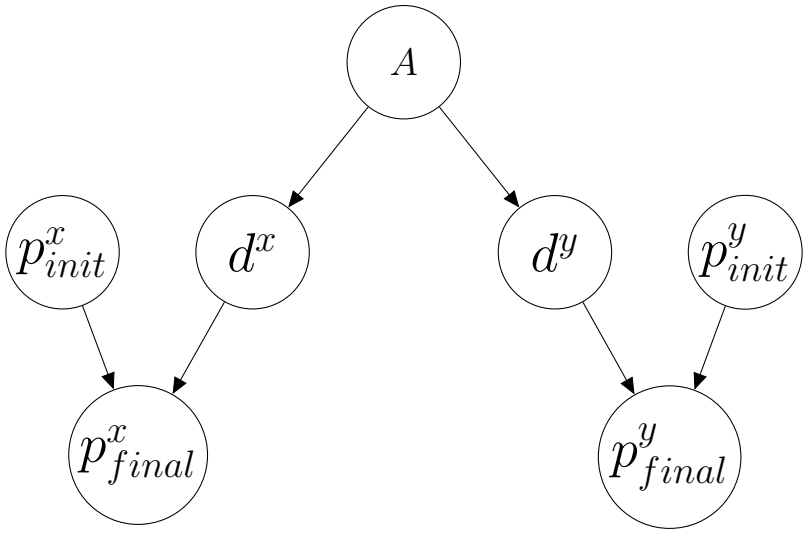


Figure 3.4: The robot learned a structural causal model using observational and interventional data across three phases. In total 120 samples were used to learn this model, including 60 during the observation phase, 40 during the validation phase, and 20 during the augmentation phase.

phase consisted of 60 sample demonstrations of a human pushing a block. During the validation and augmentation phases, we limited interventions to 20 samples per intervention. During the validation phase, the robot performed two interventions, for a total of 40 samples. The augmentation phase consisting of one intervention on the action type, consisted of 20 samples.

Figure 3.5(a) depicts the robot's performance with each tool across 5 goal locations with 5 randomly distributed initial positions per goal location. Remarkably, the top 3 best-performing tools, the short hoe ($M = 0.04, SD = 0.02$), the hoe ($M = 0.5, SD = 0.04$), and the rake ($M = 0.06, SD = 0.04$), came within a few centimeters of the goal on average, despite the relative inaccuracy of the robot and the limited set of movement directions at the robot's disposal. The tool performance roughly tracks with morphological similarity with the source tool (i.e., the hoe), with the most similar tools producing the best results (refer to Figure 3.3). The learning curve results depicted in Figure 3.5(b) tell a similar story, as the tools that are morphologically

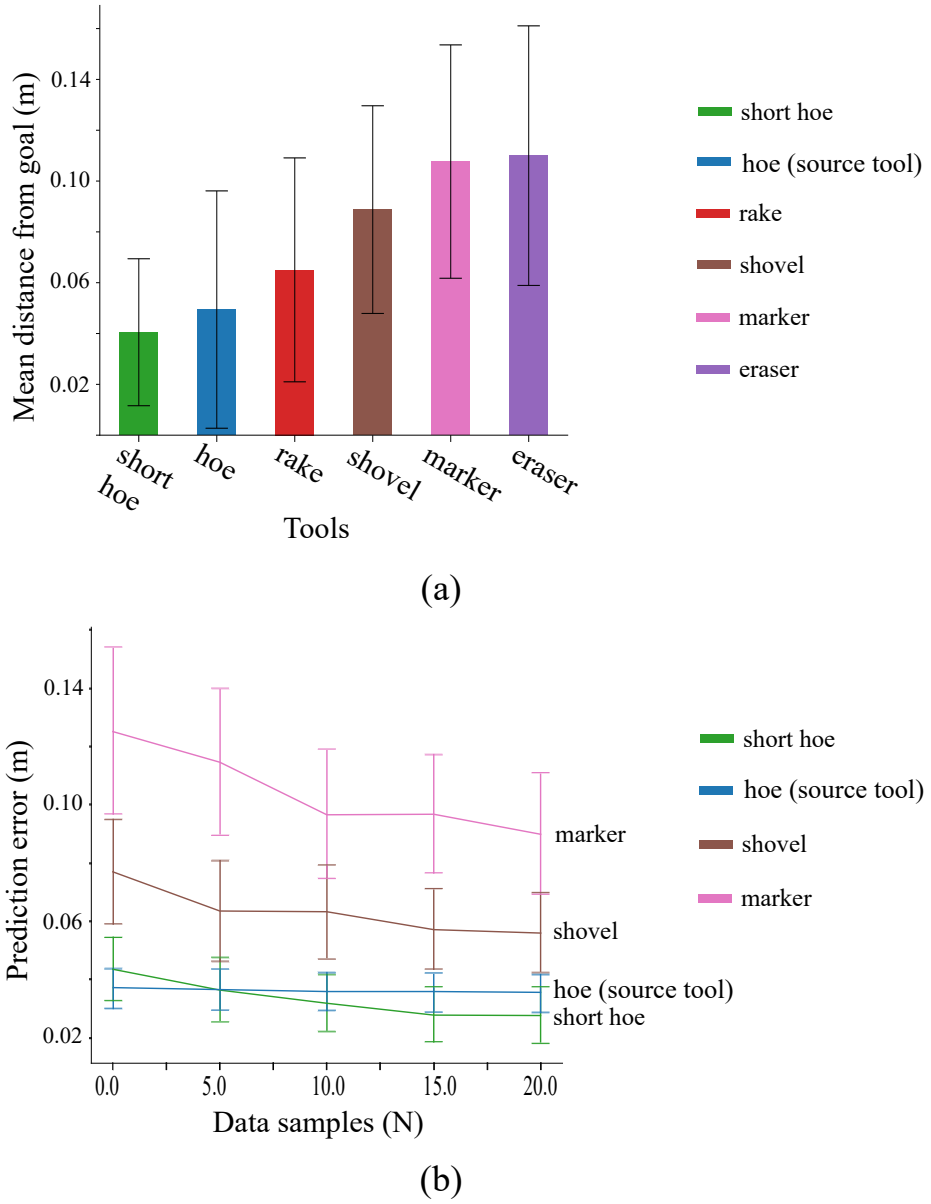


Figure 3.5: The robot was tasked with pushing a block into a goal region with a given tool. a) depicts the mean distance of the center of the block to the center of the goal region for each tool. b) depicts the learning curve for a select number of tools given initial training on the hoe. Generally speaking, usage and prediction performance degraded as a function of how morphologically distinct the testing tool was from the training tool (the hoe).

most similar to the source tool (and to a lesser extent, the shovel) benefit much more from the prior learning than the marker. Interestingly, the short hoe ultimately

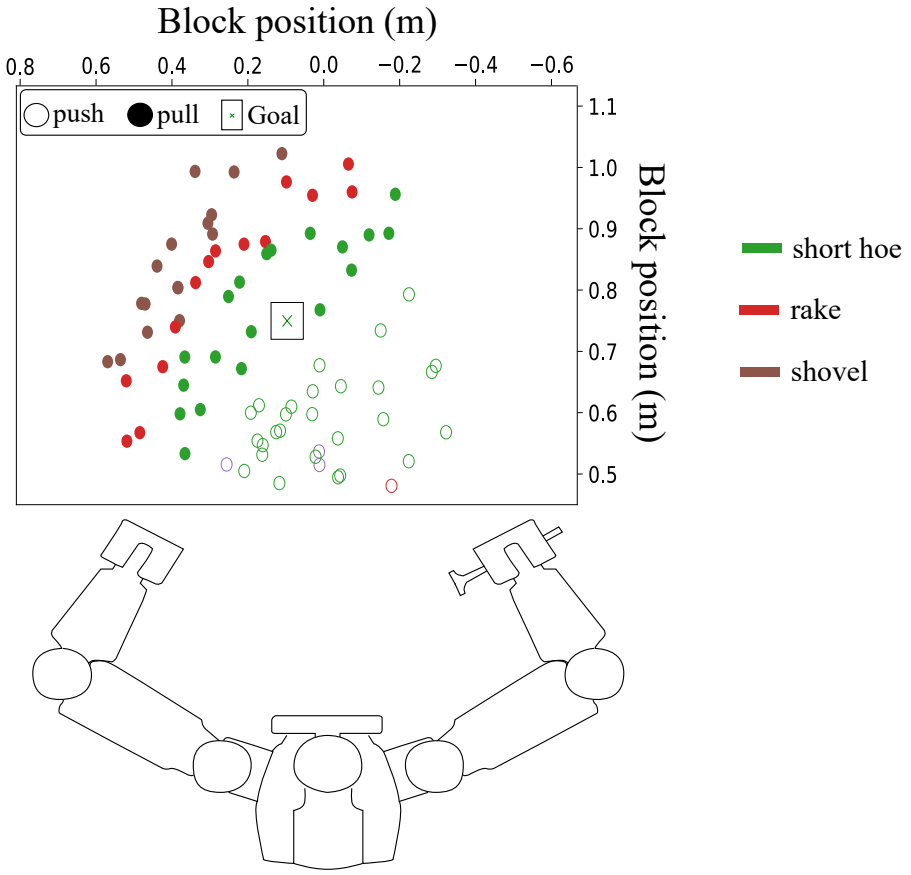


Figure 3.6: The robot was tasked with selecting the appropriate tool for manipulating a block into a static goal region. *Top* depicts the workspace, where the colored circles represents the tested initial positions of the block and the green X represents the fixed goal region. *Bottom* depicts a to-scale rendering of the Baxter robot relative to the workspace grasping a tool in its right gripper. These results indicate that the robot preferentially selected tools based on the position of the block relative to itself and the goal.

performs better than the source tool, though this is likely due to the fact that the short hoe is physically the same tool as the source tool, but just gripped closer to the tool tip, producing more consistent pushes and pulls.

Figure 3.6 depicts how our system chose tools and actions as a function of the object's initial position given a fixed goal location. Here we see action selection in line with expectations; the robot chose to push the block when the block was positioned before the goal relative to the robot and pulled it when it was positioned beyond the

goal, further supporting the notion that the system has meaningfully grounded the push and pull actions to sensorimotor effects. Figure 3.5(a) also helps shed light on the tools choices in Figure 3.6. The robot overwhelmingly preferred the short hoe for pushing and pulling, which makes sense as it is the tool the robot uses most effectively. On the periphery of the workspace, the robot begins choosing the rake for pulling, as the rake is both slightly longer than the hoe, and also the robot's next best tool. The same reasoning for the choice of the shovel on the edges of the workspace; it is the longest tool available to the robot and thus the only one capable of completing the desired pull.

3.4 Discussion

In this work, we demonstrated a method for a robot to learn and utilize a causal structural model to rapidly acquire and reason over tool affordances. The results of our experiments suggest that the grounding of actions to effects via the learned causal model enabled the robot to effectively select and use tools preferentially based on the conditions of the workspace.

We believe there are two primary advantages of this method over more common approaches to robot learning: 1) The knowledge acquired through the learning process is explicitly represented and hierarchically organized by virtue of the DAG structure of the SCM; 2) By leveraging this same DAG structure to construct neural networks, the functional mechanisms of the SCM can be learned in a relatively data-efficient way. That is, even in a dense causal network, for a given node of interest, only a subset of the nodes are required to infer the node's value, mitigating the effects of the curse of dimensionality that often plagues machine learning systems. The transparency of knowledge entailed in 1) is vital in a robotics context, for example in human-robot collaborative contexts where shared expectations amongst collaborators have been

demonstrated to be important for team success (Hayes and Scassellati, 2013).

Nevertheless, there were some limitations to this work. There were relatively few causal variables under consideration, and all of them were assumed to be observable. Currently, it is not clear how well our method would scale to learning larger, more complex causal graphs or graphs with latent causal variables. In addition, aspects of the interventional experiments conducted by the robot, including the two initial positions of the block, were hard-coded. Ideally, the robot should be able to autonomously generate its own experiments and choose values to force the interventional variables to take on. This is an important problem, as depending on the generating distribution underlying the model, some interventions may be more informative than others for uncovering causal relationships. However, we intended this work to serve primarily as a proof-of-concept of the causal learning approach described above, and to validate the notion of tool-affordance abstractions constructed using causal models. Thus, we believe the relative simplicity of the tool-use actions and the causal model to be warranted, and the development of more practical solutions to be left to future work. Indeed, in the following chapter, we will leverage many of the ideas explored here to develop a general framework for robot tool-skill learning.

However

3.5 Summary

In this chapter, we present our work enabling a robot to learn useful symbolic abstractions for tool-based reasoning. In the following chapter, we invert this learning process; we demonstrate how a robot can exploit pre-existing symbolic knowledge in order to guide the subsymbolic learning of tool-use skills.

Chapter 4

Top-down Hybrids: How Can Abstractions Benefit Tool Use Learning?[1]

In the previous chapter, we validated the notion that robots can learn causal relationships in a self-supervised way, and that these relationships can be used to construct simple, but powerful tool affordance models. In this chapter, we focus particularly on the tool-affordance models themselves; to what extent can such models be used to benefit tool-use learning and reasoning? To answer this, we developed the TRansferrIng Skilled Tool use Acquired Rapidly (TRI-STAR) framework, a top-down hybrid framework for tool skill learning from demonstration. TRI-STAR's capabilities derive from a causal-based tool affordance taxonomy. This taxonomy defines classes of tool-use tasks based on shared, invariant properties, namely their relevant cause-and-effect relations. By locating demonstrated skills within this category, TRI-STAR can tailor the way these skills are learned and reasoned about. As a consequence,

[1]Portions of this chapter were originally published as: M. Qin*, J. Brawer*, and B. Scassellati. Rapidly Learning Generalizable and Robot-Agnostic Tool-Use Skills for a Wide Range of Tasks. In *Frontiers in Robotics and AI*, 2021. (* equal contributions) (Qin et al., 2021)

TRI-STAR can not only enable a robot to learn tool-use skills quickly but to generalize them to completely novel tools and manipulated objects. These are especially important capabilities in the context of tool use, as the near limitless varieties of available tools render tool-specific skill learning impractical. TRI-STAR has three primary components: 1) the ability to learn, reason, and apply tool use to a wide variety of tasks from a minimal number of training demonstrations, 2) the ability to generalize learned skills to other tools and manipulated objects and 3) the ability to transfer learned skills to other robots.

4.1 Introduction

Imagine a robot designed to perform household chores. Such a robot will encounter many tasks requiring the use of a wide variety of tools, for example, cutting and stirring ingredients to help with cooking, scooping pet food to care for family pets, and driving screws and hammering nails to assist with house maintenance. In order to be a help and not a hindrance, such a robot would need to be capable of rapidly learning a wide assortment of tasks. In addition, given the complexity of household chores and the diverse range of objects that could be encountered, a robot should be able to generalize learned skills to novel tools and manipulated objects without needing to be retrained. Finally, one might wish to leverage learned skills from other users or transfer a library of accrued skills to a new robot without retraining.

A framework that enables such capabilities would have applications that extend far beyond the household. The search-and-rescue and disaster cleanup domains, for example, could benefit from such capabilities. Since these scenarios can be highly unpredictable and resource-limited, the robot should be able to both learn the appropriate tool-use skills rapidly and substitute learned tools for improvised alternatives. In addition, the ability to transfer learned skills to other robot platforms will en-

able rapid deployment of new models to assist or to replace a damaged teammate, regardless of different robot kinematic configurations.

This study focuses on learning and applying tool-use skills in a *task-general* manner (i.e., to handle a wide variety of tasks without predefined information for each specific task). In the course of a *task*, a single action is taken with objects in order to achieve a particular goal. The *objects* include a *tool*, an object that is "graspable, portable, manipulable, and usually rigid" (Gibson, 1979), and a *manipulandum*, an un-grasped object being manipulated by the tool. Similar to previous tool-use studies, we only consider tool-use tasks involving the following: 1) tools and manipulanda that are unjointed rigid bodies, 2) the use of contact forces to deterministically change the state of the manipulandum, and 3) a goal that can be accomplished with a single tool action, rather than a series of actions.

We report on a task-general tool use framework, called TRansferrIng Skilled Tool use Acquired Rapidly (TRI-STAR). The framework includes components such as perception, tool use learning, and tool use generalization. These components collectively endow a robot with three capabilities, or *stars*, aimed at solving challenging and commonplace problems in robot tool use. Star 1 enables basic tool use in a *task-general* manner, so that it can learn and apply a wide range of tasks with minimal training. Star 2 enables *object substitution*, the ability to generalize the tool-use skills learned with trained *source* tools (color-coded green in the figures in this chapter) by Star 1 to both novel *substitute* (color-coded blue) tools and manipulanda with no additional training. Star 3 enables *platform generalization*, the ability to transfer learned skills directly to other robot platforms.

4.1.1 Task-Oriented Approach to Tool Use

Tool-use skills are actions composed of two components: motor skills and contact poses. Motor skills concern the tool trajectory (i.e., a time series of poses of a tool) and

dynamics (i.e., the forces required for successful tool use). The contact poses consider tool–manipulandum contact poses and gripper–tool contact poses or graspings, which are dependent on the tool–manipulandum contact poses. While previous studies generally focus on one aspect of the skills, our system considers multiple skills, or the minimum set of tool-use skills that enables a robot to use a tool, which includes the tool trajectory and tool–manipulanda contact poses (henceforth referred to as contact poses).

While some tool-use studies are tool-oriented in that they seek to model tool use for a specific tool or class of tools (e.g., Stoytchev, 2005; Zech et al., 2017; Sinapov and Stoytchev, 2008), we opted for a task-oriented approach (Kokic et al., 2017; Detry et al., 2017) that learns the entire affordance model of a task. This is a more natural framing of the problem as the actions to be performed with a tool are driven by the desired effects, not by the specific tool itself. To illustrate, the actions taken with a hammer on a nail differ when one drives the nail into a board or pulls the nail out with the hammer's claw. In both tasks, the tool (the hammer) and even the manipulandum (the nail) are the same, so differences in how the tool is used can only be explained by the differences in the tasks. In a tool-oriented approach, the tool would have uniquely determined a single action for both steps.

In a task-oriented approach, effects, objects, and actions are connected through causal relationships. A desired effect causes an agent to select features of objects (e.g., the desired effect of cutting requires a tool to be sharp), and the objects and the desired effects determine the precise action to be taken (e.g., the desired position of a block determines how it should be pushed, and the size of a bowl influences the radius of a stirring motion). While these objects-effects relations, actions-effects relations, and objects-actions relations, respectively, may differ across tasks, they remain constant across instances of a particular task and are useful when learning and generalizing tool use.

Specifying these three relations for each task is impractical and learning these relations for each task can be data intensive. However, the causal structure of this approach implies that tasks with similar desired effects also share common features. Therefore, we compiled an affordance taxonomy that categorizes tasks based on their effects with respect to manipulanda (For details, see Section 4.2.2). The advantage of utilizing taxonomic knowledge is that information does not need to be manually specified for new tasks either when learning a task or applying the learned tool use skills. In this way, taxonomic knowledge can help to reduce the training data needed.

4.1.2 Star 1: Learning and Applying Task-General Tool Use

Star 1 is the ability to perform basic tool use, that is, to learn and apply the actions-effects relations in the affordance model using a source tool and manipulandum. This includes determining the contact poses and the course of actions depending on the effects. In this section, we first describe relevant studies. We then describe the challenges in learning basic tool use and briefly describe the tests we conducted.

Studies focusing on modeling motor skills often ignored contact poses, though they have been applied to tool-use tasks such as swinging tennis rackets (Ijspeert et al., 2002), batting (Peters and Schaal, 2006), playing ball-in-a-cup (Kober et al., 2008) or table tennis (Muelling et al., 2010), pouring (Pastor et al., 2009; Rozo et al., 2013), writing letters (Lioutikov et al., 2017) or digits (Droniou et al., 2014), and peg-hole insertion (Gao and Tedrake, 2021) with methods such as dynamical movement primitives (Schaal, 2006; Ijspeert et al., 2013) or probabilistic movement primitives (Paraschos et al., 2013). In one study, experimenters hard-coded the contact poses that the end of a peg should align with the top of a hole vertically when learning the peg-hole insertion task (Gao and Tedrake, 2021).

Studies that did not ignore contact poses (Kemp and Edsinger, 2006; Hoffmann et al., 2014) typically utilize the tool tip as a simplified representation of the contact

area. Yet, in practice, the contact area can comprise any arbitrary area at any location on a tool, such as the tip of a screwdriver, the blade of a knife, the face of a hammer, or the concave surface of a ladle. Moreover, with such a simplification, the relation between the tool and manipulandum is reduced to the angle of contact, which is insufficient for tasks like screw-driving; a screwdriver should contact a screw not only perpendicular to the head of the screw but also with the correct rotation about the tip axis. Additionally, such simplified representations cannot account for tasks that may have multiple viable contact poses; a hammer may approach a nail from infinitely many orientations about the head axis of the nail and thus have an infinite number of viable contact poses.

While the aforementioned studies did not incorporate causal relations, studies that focused on these relations did not consider action generation. Sinapov and Stoytchev (2007) and Stoytchev (2008) learned how predefined linear end-effector trajectories of different tools lead to positional changes of a manipulandum. Zech et al. (2017) attempted to learn relationships between effects and contact poses to aid in tool selection but predefined a contact pose template. Other studies (Gonçalves et al., 2014b,a; Dehban et al., 2016; Moldovan et al., 2013) learned these relations from a probabilistic approach but also with predefined end-effector trajectories.

Star 1 utilizes an affordance model of tasks to guide the learning of tool-use skills. We demonstrated seven tasks (knocking, stirring, pushing, scooping, cutting, writing, and screw-driving) that were learned with a small number of training samples and tested different types of tool use. This range of tasks tested the learning and application of tool use given different task types, such as stirring, screw-driving, and pushing, each corresponding to a type defined in the taxonomy. When learning tool-use skills, we consider the minimum set of tool use that enables a robot to use a tool, which includes the tool trajectory and tool-manipulanda contact poses (henceforth referred to as contact poses).

4.1.3 Star 2: Task-General Object Substitution

Star 2 is the ability to generalize learned tool-use skills from source to substitute tools or manipulanda that can complete the task, including objects that share a common geometric template (e.g., mugs differing in shape and size as in Brandi et al. (2014)), or share no common form-factor (geometrically-distinct objects, e.g., pushing an object with a cake-cutter rather than a toy rake). To generate actions, an object-substitution algorithm must adjust learned trajectories for tasks such as stirring in a smaller container and produce the appropriate contact poses. The contact poses for many tasks can be obtained by finding the alignment between the source and substitute objects based on features for tasks such as cutting, but for some tasks (like pushing) the contact poses are determined by desired effects of the tasks. Similar to previous tool use studies, we focused on geometric features only.

Many previous studies employed task-specific approaches that limited the robot's ability to transfer to tools that share common form factors. Some of these approaches required hand-engineered information to find an alignment for each task (e.g., Brandi et al., 2014; Stückler and Behnke, 2014; Hillenbrand and Roa, 2012). Providing hand-engineered information for each task exhibits two disadvantages. First, algorithms requiring hand-engineered information constrain their user-friendliness for naïve end-users who lack the knowledge to train these algorithms adequately. Second, engineering information for each task is time-consuming and impractical in real-world settings requiring the use of many tools.

Other approaches that can accommodate tools of various shapes usually require prohibitively large amounts of data per task. For example, over 20,000 training examples were needed to learn and generalize in the pushing task (Xie et al., 2019); 18,000 simulated tools were used to generalize tool use in a sweeping and nail-hammering task (Fang et al., 2020); 5,000 vectorized representation of tools were used to train a neural network to generalize tool use in the scraping, cutting and scooping tasks

(Gajewski et al., 2019; Abelha and Guerin, 2017). Acquiring a large training sample set is infeasible when tasks need to be learned rapidly or when many tasks need to be learned. Moreover, these studies only considered tool substitutions but not manipulandum substitutions limiting their applicability to many real-life tool use applications.

Star 2 performs object substitution by adjusting tool use skills learned by Star 1 using all three relations comprising taxonomic knowledge without additional training. While the actions-effects relations assisted the generation of actions to different task configurations in the same way as in Star 1, the two object-related relations help to generate contact poses and adjust learned trajectories. This ability to adapt trajectories to accommodate substitute objects, as well as the ability to perform tool and manipulandum substitution are two advantages of our approach that are not typically considered in other studies. We evaluated Star 2 with five tasks (knocking, stirring, pushing, scooping, and cutting). The substitute objects differed from the source objects in size, shape, or a combination of both. We also tested trajectories requiring adjustments based on geometric features of the manipulanda (e.g., stirring and cutting), desired effects (e.g., pushing), and trajectories requiring no adjustments (e.g., hammering).

4.1.4 Star 3: Transferring Tool Use Skills to Other Robot Platforms

Star 3 is the ability to transfer tool use to other robot platforms. This requires a robot-independent representation of tool use. Though learning trajectories and dynamics in the joint state space is common in learning motor skills, such representations make it challenging to transfer learned skills to robots with different hardware configurations. Learning in the Cartesian space is more conducive to cross-platform transfer, though it suffers from practical limitations.

When learning in Cartesian space, prior tool-use studies (e.g., Xie et al., 2019; Fitzgerald et al., 2019) used the gripper pose as a proxy for the tool pose to simplify the perception problem. In these studies, rather than learning tool-manipulandum contact poses and tool trajectories, the gripper-manipulandum relative pose and gripper trajectories were used to learn tool use. Using gripper poses assumes that the grasps of a tool remain consistent across training and testing regimes, which is difficult to ensure outside of a controlled lab setting even on the same model of robot. When such assumptions cannot be met and a robot needs to grasp a tool differently, workarounds sometimes are employed such as treating learned tools as novel (Sinapov and Stoytchev, 2008; Mar et al., 2017), which complicates the skill transfer process.

In contrast, a trajectory of a tool, rather than an end-effector, is a flexible and direct representation for tool actions. Such a representation is not tied to any particular robot configuration and does not require grasping consistency within or across platforms. This flexibility enables a robot to perform tool use with different grasps of the same tool. Crucially, this flexibility also extends to transferring skills to other robot platforms.

Star 3 performs tool use transfer from a source robot to a substitute robot by leveraging our platform-agnostic representation of tool-use skills. The strength of using such a representation is that it updates a common representational schema (i.e., Cartesian end-effector trajectories) in a simple way, but nevertheless greatly impacts the flexibility and generalizability of tool skills. The process of applying the skills is otherwise the same as in Star 1. We tested the transfer of skills learned with a Universal Robotics UR5e arm to both a Baxter robot and a Kuka youBot robot with six tasks (knocking, stirring, pushing, scooping, cutting, and writing). These three robots have different degrees of freedom (DoF) and are kinematically distinct. UR5e has 6 DoF, and one arm of Baxter has 7 DoF, which allows the robot to pose its end-effector freely in the 3D space. YouBot without the mobile base has only 5

DoF and thus limits the robot's ability to reach arbitrary poses. Depending on initial conditions, a robot might abort execution or slightly adjust a trajectory if it cannot be fully executed.

4.2 Methods

The TRI-STAR framework focuses on learning geometrically-based tool use via Learning from Demonstration with position control[2]. We first introduce and summarize the representational schemas we use throughout the system which include the affordance taxonomy, trajectory and contact pose-based tool skills, and our 3D model and 6D pose-based object representation. Subsequently, we detail the three primary capabilities of our system.

4.2.1 Preliminaries

The following terms are used in this method section. Pose describes the translational and rotational state of a 3D rigid body, which is generally represented as a rigid homogeneous transformation matrix in the Lie group $SE(3)$ or exponential representation parameterized as a *screw axis* and an *angle*, which is a concept from screw motion. By the Mozzi-Chasles' theorem (Chasles, 1830), for any arbitrary pose change or transformation, an equivalent screw motion always exists. A screw axis S is a normalized twist (i.e., a spatial velocity) which is a six-dimensional vector consisting of the axis ω of the helix and a linear velocity v at the origin:

$$S = \begin{bmatrix} \omega \\ v \end{bmatrix} \in \mathbb{R}^6$$

More information about the use of exponential representations in robotics can be

[2]Source code is available at https://github.com/ScazLab/Frontiers_Robot_Tool_Use.git

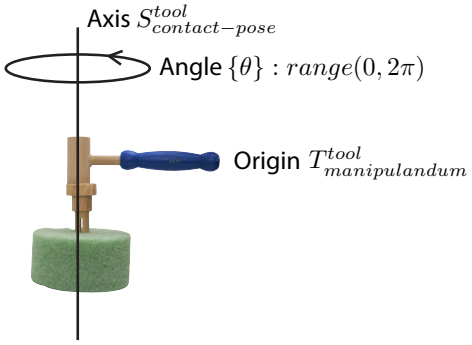
found in Lynch and Park (2017) and Siciliano and Khatib (2016). Screw axes offer a flexible and compact representation of actions. They provide a qualitative description of 3D motion by characterizing the shape of a trajectory such as straight or curved lines, circles, and helices, while the angles quantify the length that a trajectory should follow along the shape defined by the screw axis. As a result, the motion primitives are represented with screw axes in this study.

The *frame of reference*, or *reference frame*, is the default coordinate system in which a pose is referenced. While this is a standard definition, it is different from previous studies (e.g., Sinapov and Stoytchev, 2007), where the term is used to describe an object of interest. To illustrate the differences, when we refer to things like the "manipulandum frame," for example, we mean the the coordinate system with the pose of the manipulandum at the origin, while the object of interests could be anything rather than restricted to the manipulandum as in previous studies.

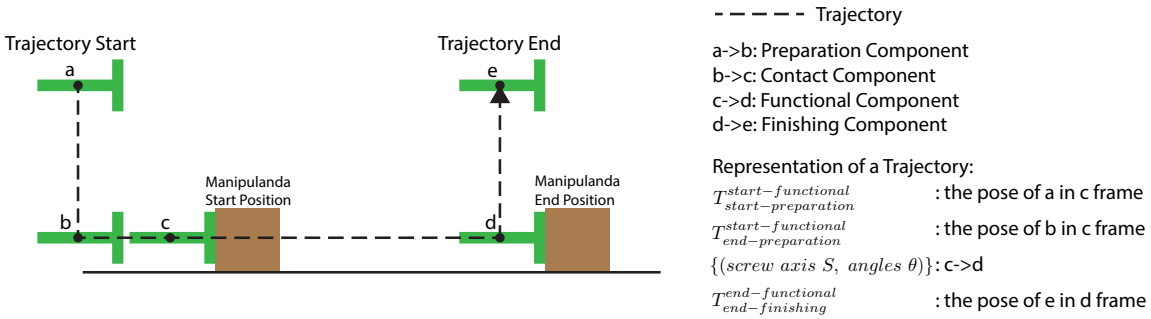
4.2.2 Representations

Actions Representation: Trajectory and Contact Poses

A trajectory consists of four components as shown in Figure 4.1a: 1) the preparation component, which brings the tool in close proximity to the manipulandum, 2) the contact component which initiates contact with the manipulandum, 3) the functional component which acts on the manipulandum, and 4) the finishing component, which moves the tool away from the manipulandum, terminating the trajectory. The main part of the trajectory is the functional component. We represent this component using screw axis representations which are compact and easily adapted for tool use. Though we also included other components, we consider such components peripheral to the tool skill proper and thus are not the focus of this study. Keeping with other tool use studies that either completely ignore such components or hard-code them (e.g., Sukhoy et al., 2012), we represented these components simply using trajectory



(a) Parametrization of a Class of Contact Poses.



(b) Components of a Trajectory and its Parametrization.

Figure 4.1: Star 1 learns action representations using tool trajectories and contact poses. (a) depicts the parametrization of a contact pose using a nail-hammering task as an example. (b) depicts the four component trajectories that comprise a hypothetical demonstration of a pushing task.

endpoints.

We represent the functional components with a series of segments $\{(screw\ axis\ S,\ angles\ \theta)\}$ with each segment parameterized with exponential representations of a pose change. The advantage of such representation is twofold. First, since the screw axis includes all six DoF, no coupling between dimensions is needed as in previous methods (Schaal, 2006; Ijspeert et al., 2013; Paraschos et al., 2013). Second, in accordance with other representation schemes, trajectories can also be easily rescaled and rotated. Such a representation may not be ideal for other robot manipulation tasks such as pick and place where learned trajectories are flexibly warped based on different start and goal poses. However, this representation is suitable for the tool

use domain where trajectories may need to be warped in a structured way based on taxonomic knowledge (e.g., to adapt a learned straight trajectory to push along a curved one required by the desired effects) or extended along the shape outlined by the screw axis such as when driving the same screw into boards of different thicknesses.

Contact poses are represented with equivalence classes of poses, $\{T_{man}^{tool}\}$, that treats all poses formed from rotating around some axis as being equivalent. This is a uniform representation for finite contact poses such as driving screws and infinite contact poses such as nail-hammering. Each element T_{man}^{tool} is a manipulandum pose in the tool frame (i.e., the tool frame is the pose of the tool when initiating contact with the manipulanda). Such a representation is able to accommodate contact areas of any shape located anywhere on a tool and manipulandum as well as represent any orientation between the two objects. The transformations in the same class can be obtained by rotating about an axis S_{cp}^{tool}. As a result, a class of contact poses (shown in Figure 4.1b) is parameterized as an axis S_{cp}^{tool}, a transformation T_{man}^{tool} as the origin, and a group of angles θ such that a viable contact pose can be obtained by rotating an angle θ about the axis S_{cp}^{tool} starting from T_{man}^{tool}. In this way, this class can represent a unique contact pose (i.e., a unique angle which is zero), limited contact poses (i.e., a limited number of angles), or an infinite number of contact poses (i.e., the angles within a range).

Object Representation: 3D Models and 6D Poses

TRI-STAR is designed for a robot to utilize novel tools without prior training. In order to accomplish this the algorithm requires the robot to obtain 3D models of the novel objects under consideration. This is because many of TRI-STAR's capabilities depend on the ability to find good alignments between object point clouds (see, for example Sec. **??**). We used Microsoft Azure RGB-D cameras, which are commonly used and relatively inexpensive sensors, to obtain raw partial 3D point clouds. With

the relatively low fidelity of perceived partial point clouds, available methods could not obtain full 3D models of sufficiently good quality. Therefore, it was necessary to design a pipeline to ensure the fidelity of 3D tool and manipulanda models.

This pipeline begins with first mounting an object in the robot's end-effector. The robot can then rotate its end-effector around an arbitrary axis to ensure the object's back and front are visible to the 3D camera. A series of raw point clouds are obtained while the robot steps through the trajectory. The background is then pruned to obtain the partial point clouds of the objects. Given the pose of the end-effector at each step, the partial point clouds are merged by transforming these point clouds to the initial pose. To account for noise, we optimize the rotation axis represented as a screw axis S with the Quasi-Newton method (Dennis and Moré, 1977) by minimizing the sum of the Euclidean distances between the bounding boxes of the partial point clouds and the bounding box of the merged point cloud. As parts of the objects are occluded by the robot's own gripper, the robot obtains two such merged scans and registers them to create the final complete scan. Supplemental scans using Autodesk Recap[3] photogrammetry software was also used to obtain point clouds for objects that are challenging for the robot to grasp. Although we attempted to design the entire process to be autonomous, the grasping during scanning and tool use requires an experimenter to assist with mounting an object to the gripper.

To obtain smoothed triangle meshes, the models are post-processed automatically with a script using meshlabxml[4], a python interface to MeshLab[5], similar to a previous study (Gajewski et al., 2019). The point clouds are upsampled with Poisson-disk sampling with input 5,000, meshed with Ball-Pivoting, smoothed with Taubin smoothing, and holes are filled with the default settings. The meshes are then centralized and realigned based on their minimum bounding boxes.

[3]https://www.autodesk.com/
[4]https://github.com/3DLIRIOUS/MeshLabXML
[5]https://www.meshlab.net/

We used a non-marker-based perception system and estimated the pose of the objects from raw sensor input. Two Azure devices are placed on the two sides of the workspace to capture a complete point cloud representation of the workspace. Background and foreground point clouds are retrieved from both sensors. The workspace is isolated, and the desktop is removed with random sample consensus (RANSAC; Fischler and Bolles, 1981) from these point clouds. To obtain a partial point cloud of the manipulanda, the background is subtracted from the foreground point clouds. The object's pose in the world frame T_{man}^{world} is obtained by rigid registration between the partial point cloud and the full 3D model. The pose with a higher fitting score, measured by calculating the ratio of inlier point correspondences over the total number of target points, is chosen. If the scores from both sensors are similar, the averaged pose is used.

The method of obtaining tool poses in the end-effector frame is similar to the method above, except for the extra step of removing points belonging to the gripper to isolate the tool. The pose of the tool in the end-effector frame is then obtained with $T_{tool}^{ee} = (T_{ee}^{world})^{-1} \times T_{tool}^{world}$ where $\times$ is matrix multiplication, and the superscript -1 represents matrix inversion, given the perceived pose of the end-effector in the world frame T_{ee}^{world} and the perceived tool pose T_{tool}^{world}. Similar to previous tool use studies, we assume a fixed grasp for a tool once it is in the robot's end-effector.

Task Representation: Affordance Taxonomy

We developed a taxonomy of affordances to assist learning and application of tool use. Our taxonomy (Figure 4.2) recognizes two fundamental task types using desired effects referenced in the world frame. We focus on tasks with effects described by pose changes of the manipulandum as they can be easily perceived. Since the goals for everyday tool use tasks generally require simple motions of the manipulanda, one screw axis can be used to characterize the shape of an effect-directed motion primitive.

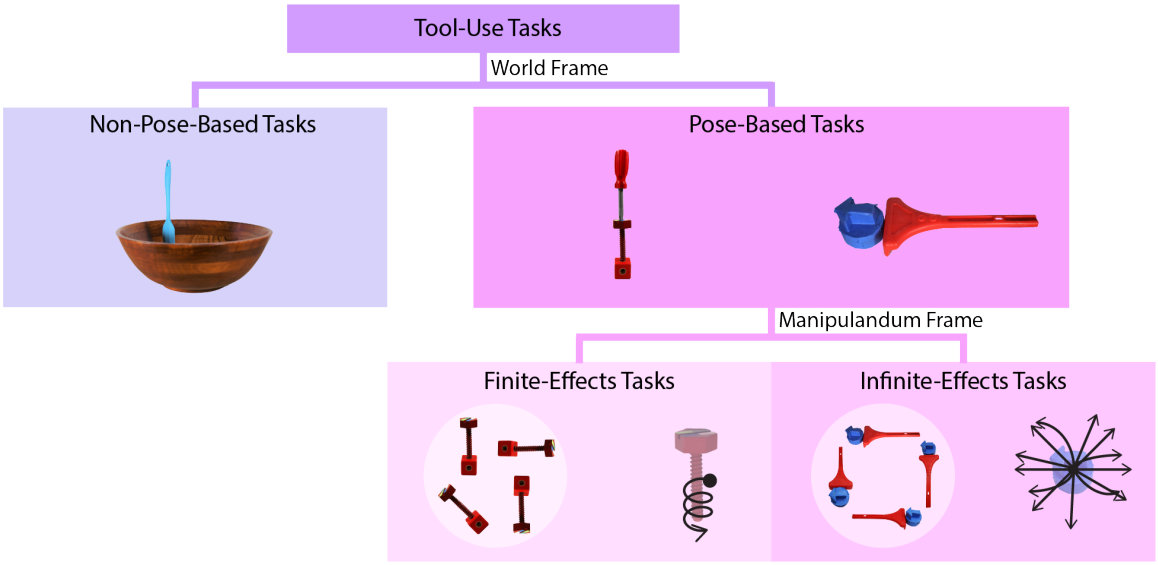


Figure 4.2: Tool use tasks can be organized taxonomically based on their respective affordances. The taxonomic structure emerges when observing effect-based motion primitives from different frames of reference. The effects of Non-Pose-Based Tasks do not involve changes in the manipulanda's poses, which is different from Pose-Based Tasks. For Pose-Based Tasks, there are infinitely many possible pose changes of the manipulanda in the world frame given different start poses. However, there are only finite possibilities for Finite-Effects Tasks when the effects are considered in the manipulanda frame. For example, a screw may have very different end poses in the world frame when given different start poses. However, when referenced by itself, screwdriving produces only one observable effect, namely to move the screw in the direction of its tip. In contrast, Infinite-Effects Tasks have infinitely many effects in both the world and the manipulanda frame. Learning different subtypes in the taxonomy requires different causal relations. For example, the tool-manipulanda contact poses of the Finite-Effects Tasks are not determined by the desired effects but by the features of the objects; one can determine how a screwdriver should contact a screw without providing the desired effects. For Infinite-Effects Tasks, the tool-manipulanda contact poses are determined by both object features and the desired effects; one should be provided with the desired location of an object before determining how a stick should contact the object to push it.

Non-Pose-Based Tasks are tasks with zero screw axes, representing the case where the pose of a manipulandum (e.g., a bowl) is not changed as a result of the tool usage (e.g., stirring liquid in the bowl) in the world frame. Pose-Based Tasks are tasks with non-zero screw axes such that the pose of a manipulandum changes as a result of tool use, though two further subdivisions emerge when observing in the manipulandum

frame. Finite-Effects Tasks such as screw-driving are tasks where a unique screw axis in the manipulandum frame exists to describe actions-effects relations while there are still infinitely many effects in the world frame. Infinite-Effects Tasks, in contrast, like pushing a toy with a rake to the desired location, have infinitely many screw axes in the manipulandum frame to represent effects.

The taxonomy characterizes what causal relations should be considered in the subtypes. Learning tool use in Star 1 is consistent in all three types of tasks, except that Infinite-Effects Tasks require modifications based on the actions-effects relations. The contact pose required when completing a pushing task, for example, depends heavily on the goal pose of the manipulandum. Applying tool use, in contrast, requires different task-specific information depending on the task type:

- The actions-effects relations specify how actions should be updated based on the desired effects, which are crucial for Pose-Based Tasks because they determine, for example, the length of the trajectory when driving a screw into a thick piece of wood versus a thinner piece.

- The objects-effects relations specify which object features are relevant for achieving the desired effect such as a sharp edge of a tool in the case of a cutting task. These relations are important for generating contact poses for object substitution for all tasks except the Infinite-Effects Tasks. These tasks do not require manipulanda-effects relations but require actions-effects relations since the contact pose depends on, for example, where the manipulanda should be pushed to in a pushing task.

- The object-action relations dictate how the actions should be updated based on different object features, which are relevant for the Non-Pose-Based Tasks such as stirring where the radius of a stirring trajectory is dependent on the size of the container containing the mixture being stirred.

These relations capture common features across tasks within each category of the taxonomy and can be used to guide the learning, application, and transferring of tool use to substitute objects.

4.2.3 Star 1: Learning and Applying Task-General Tool Use Skills

In Star 1, our framework first categorizes task demonstrations using our taxonomy. The taxonomic knowledge of the identified category is then used to learn tool use skill representation (i.e., the contact poses and trajectories) and generate actions with desired effects not seen in the training samples. In the following sections, we describe how tool use is learned and applied to novel task configurations. Specifically, we first detail the simulated demonstrations used to train the skills evaluated in this study. Subsequently, we discuss how demonstrations are categorized using our affordance taxonomy and how the corresponding taxonomic knowledge is leveraged to learn trajectory and contact pose representations. Next, we detail how the system utilizes new task configurations to apply learned skills by generating new trajectories and contact poses.

Learning Tool Use Skills

The input data required by our algorithm includes the start and goal pose of the manipulanda in the world frame and the tool trajectories as the keyframes in the world frame. Twenty simulated training samples per task were provided. Training samples were obtained with kinematic teaching of keyframe demonstrations in simulation. Each sample was a single demonstration of a task using a source tool and manipulandum. The samples were assumed to be successful demonstrations of a task, as no sophisticated outlier removal methods were utilized.

With the start and goal poses of the manipulanda, the system can infer the cate-

gory of task being demonstrated to be used to guide the learning of trajectories and contact poses. If the effects of all demonstrations are zero vectors, then this task is a Non-Pose-Based Task. Otherwise, it is a Pose-Based Task. If it is the latter, the effects are converted to the manipulandum frame (i.e., the manipulanda frame is the start pose of the manipulanda) and are clustered based on the Euclidean distance between ω parts and the Euclidean distance between v parts of sample screw axes. If a unique cluster is found, then this task is considered a Finite-Effects Task. Otherwise, it is an Infinite-Effects Task.

The trajectory between two adjacent keyframes in a given demonstration is assumed to be interpolated, which may or may not be linear depending on rotational differences between the two frames. The keyframes can include only the start and goal pose of segments or any arbitrary number of midpoints. The keyframes are first merged into segments automatically. The different components of the trajectory are then identified by the framework. However, each component is assumed to have the same shape across demonstrations except for the functional component. Given a demonstrated trajectory comprised of keyframes, the framework first groups the keyframes into segments with similar transformations between keyframes (i.e., the grouping stage). A component might be missing for different types of tasks, which is identified during this grouping stage. Subsequently, each segment, or partial segment, is then parameterized with the appropriate component, and represented with $T^{start-func}_{start-prep}$, $T^{start-func}_{end-prep}$, $\{(screw\ axis\ S,\ angles\ \theta)\}$, and $T^{end-func}_{end-fin}$, as illustrated in Figure 4.1a (i.e., the parametrization stage).

The first step in the grouping stage is to identify the preparation component and the finishing component, which is used to find the start pose of the preparation component $T^{world}_{start-prep}$, goal pose $T^{world}_{end-prep}$ of the preparation component (it is also the start of the contact component $T^{world}_{start-con}$), the start pose of the finishing component $T^{world}_{start-fin}$ (it is also the end of the functional component $T^{world}_{end-func}$), and end pose of

the finishing component $T^{world}_{end-fin}$. To do this, the transformations between keyframes in the world frame are converted to the screw motion representation. Adjacent transformations with similar screw axes are merged. The similarity is evaluated using the Euclidean distance between ω parts and the Euclidean distance between v parts of sample screw axes. The merging is done by averaging the screw axis and summing the angles. After merging, the first segment is assumed to be the preparation component, while the last is assumed to be the finishing component. The start and end poses of these components can thus be found.

The second step in the grouping stage is to identify the other components. For Non-Pose-Based Tasks, the rest of the segments are assumed to be the functional component, and the contact component of this type of task is assumed to be a segment with no transformations. For Pose-Based Tasks, the contact poses are assumed to be unchanged once the tool contacts the manipulanda. Therefore, the start of the functional component $T^{world}_{start-func}$ (which is also the end of the contact component $T^{world}_{end-con}$) can be obtained with $T^{world}_{start-man} \times (T^{world}_{end-man})^{-1} \times T^{world}_{end-fin}$. Since the start (i.e., the end of the preparation component) and end (i.e., the start of the functional component) poses of the contact component are known, the contact component is found by interpolating these poses. This is obtained by calculating the screw axis of the transformation between the start and end pose and sampling angles with 1-degree intervals. Although the start and end pose of the functional component is known, the functional component is not a simple interpolation as it may need to follow a specific trajectory. Therefore, the algorithm allocates the remaining segments to the functional component, after excluding the partial segment belonging to the contact component. The partial segment is found by identifying the overlap between the first proceeding segment of the preparation component and the contact component.

In the parametrization stage, the keyframes are converted to different reference frames for easy application. The start and end pose of the preparation components

are converted to the frame of the start pose of the functional component, resulting in $T_{start-prep}^{start-func}$ and $T_{end-prep}^{start-func}$, respectively. The end pose of the finishing component is converted to the frame of the end pose of the functional component, which is $T_{end-fin}^{end-func}$. If multiple segments comprise the functional component, each segment is represented with screw motion and the start pose of this segment is used as the reference frame. As a result, the trajectory of a demonstration is represented using $T_{start-prep}^{start-func}$, $T_{end-prep}^{start-func}$, $\{(screw\ axis\ S,\ angles\ \theta)\}$, and $T_{end-fin}^{end-func}$.

The next step of the parametrization stage is to find a template from all the training samples. The functional components of Infinite-Effects Tasks are ignored, as they are determined by the desired effects, rather than a shared trajectory template. For the rest of the tasks, the number of segments comprising the functional component should be the same for each task. For the minority of demonstrations that are inconsistent with the number of segments that the majority of the demonstrations are associated with, those samples are excluded. For the remaining valid training samples, each segment of the component derived from different demonstrations is averaged. The transformations $T_{start-prep}^{start-func}$, $T_{end-prep}^{start-func}$, and $T_{end-fin}^{end-func}$ are also averaged from each demonstration.

Using the method described above, the trajectories can be learned. We now shift our attention to contact pose learning. Our current algorithm assumes a single contact area on the source tool when performing a particular task, though this could be relaxed in future studies. The contact area of the tool and the manipulandum were determined by proximity. For Infinite-Effects Tasks like object pushing where task success is contingent on the goals of the manipulandum, a change-of-basis of the start pose of the manipulandum is performed in order to incorporate the goal into its representation so that the contact poses are effect-based. The demonstrated contact poses are then converted to our representation of a class of contact poses using S_{cp}^{tool}, T_{man}^{tool}, and a group of $\{\theta\}$.

For Infinite-Effects Tasks, we perform a change-of-basis on the start poses of the manipulandum before calculating the contact poses in order to account for the goal-directed nature of these tasks. The x axis is chosen to be the moving direction of the manipulanda, which is the normalized v part of a screw axis representing the transformation of the manipulandum from the start to the goal in the world frame. The z axis is chosen to be the direction of standard gravity. If the x axis and z axis are parallel, an arbitrary direction is chosen ahead of time which is not parallel to the standard gravity. The y axis is obtained with the right-hand rule, which is the cross product of x and z. To ensure the perpendicularity between x and z, the z axis is recalculated with the cross product of x and y. The position of the manipulanda remained to be the perceived position.

The contact pose of each demonstration T^{tool}_{man} are obtained by $(T^{world}_{start-func})^{-1} \times T^{world}_{start-man}$ where $T^{world}_{start-func}$ is the tool pose at the start of the functional component and $T^{world}_{start-man}$ is the start pose of the manipulanda. Then the contact poses from each demonstration are converted to our representation, a class of contact poses. The axis between any two contact poses is calculated, and the poses whose axis deviate too much from the majority of axes are excluded. An arbitrary pose, generally the pose of the first demonstration, is chosen as the origin T^{tool}_{man}. The transformations between valid contact poses and this origin are calculated in the origin frame and represented using screw motion. The averaged axis S^{tool}_{cp} is used as the axis of this class. For the angles obtained, if the Kolmogorov-Smirnov test (Daniel, 1990) on the group of angles showed no significant difference from a uniform distribution, then the range of this angle is used to represent $\{\theta\}$. Otherwise, the groups of angles are clustered using density-based spatial clustering of applications with noise (DBSCAN; Ester et al., 1996), and the mean of each cluster is included in $\{\theta\}$.

Applying Tool Use Skills

To apply the learned tool-use skills with the source tool and manipulanda, configurations of a task should be provided. These include the start $T^{world}_{start-man}$ and goal pose $T^{world}_{goal-man}$ of the manipulandum $T^{world}_{goal-man}$. The goal pose $T^{world}_{goal-man}$ can be provided by perception (e.g., placed at the desired location) or by the experimenter in the form of a transformation matrix. The start pose $T^{world}_{start-man}$ is always perceived. The goal is always assumed valid for the given task and could be achieved by the given tool.

To use a tool, the contact poses and tool trajectories need to be found. The contact poses are generated based on learned contact poses and taxonomic knowledge. Since multiple possible contact poses T^{tool}_{man} exist for each task, multiple corresponding tool trajectories are generated. These tool trajectories are then converted into end-effector trajectories to be executed by the robot given the current perceived tool grasping pose. Trajectories are considered candidates if their functional components can be executed since the complete execution of the functional component is crucial to perform a task. The final trajectory is chosen from the candidates that minimize the required joint changes. If none of the functional components can be executed in full, the robot simply aborts execution. Otherwise, the robot attempts to execute as many components or partial components as possible since the full execution of other components is not central to successfully completing the task.

Now we consider how to generate a trajectory. Given a contact pose T^{tool}_{man} obtained from above, which is equivalent to $T^{start-func}_{start-man}$, and the start $T^{world}_{start-man}$ and goal pose $T^{world}_{goal-man}$ of a manipulandum, the start $T^{start-man}_{start-prep}$ and end $T^{start-man}_{end-prep}$ of the preparation component in the manipulandum frame is calculated using the learned trajectories by $(T^{start-func}_{start-man})^{-1} \times T^{start-func}_{start-prep}$ and $(T^{start-func}_{start-man})^{-1} \times T^{start-func}_{end-prep}$ (the information from the learned trajectories are labeled with an enclosed rectangle), respectively. The preparation component in the manipulandum frame is then found by finding the interpolation between its start and end pose. The contact component

in the manipulandum frame is obtained using the same method, with its start pose being the end pose of the preparation component and the end pose being the start of the functional component. In terms of the functional component, each segment of $\{(screw\ axis\ S,\ angles\ \theta)\}$ is found by interpolating the learned trajectory and converting those transformations to the manipulandum frame for the Non-Pose-Based Task. For Finite-Effects Tasks, the length of the trajectory, which is the angle in the screw motion representation, is adjusted according to the goal while the learned shape described by the screw axis remains the same. For Infinite-Effects Tasks (e.g., pushing), both the shape and length are determined by the goal with the end pose of the functional component being $(T^{world}_{start-man})^{-1} \times T^{world}_{end-man} \times (T^{start-func}_{start-man})^{-1}$, and the trajectory is found by interpolating the start and end pose. The end pose of the finishing component is calculated using the learned trajectory as $T^{start-man}_{end-func} \times T^{end-func}_{end-fin}$. In the end, each pose in the trajectory $T^{start-man}_{tool}$ is converted to the world frame with $T^{world}_{start-man} \times T^{start-man}_{tool}$. In the writing task introduced in Section 4.3, when a different scale of the trajectory (e.g., write a larger or smaller "R") is requested, the angle θ in $\{(screw\ axis\ S,\ angles\ \theta)\}$ is scaled if the screw axis represents translational changes only, otherwise the v part of the S in $\{(screw\ axis\ S,\ angles\ \theta)\}$ is scaled. This works because the screw axis is in the previous pose's frame, and the v represents the velocity at the origin. To rotate the trajectory (e.g., to produce a tilted "R"), one can simply rotate $T^{world}_{start-func}$. The corresponding start and end pose of other components need to be updated accordingly.

We now shift our attention to generating contact poses. For the learned class of contact pose whose $\{\theta\}$ is composed of discrete values, the contact pose in the matrix form corresponding to each value is calculated. If $\{\theta\}$ is a range, the contact poses are treated as discrete values by sampling angles from the range by 1-degree intervals. For Pose-Based Tasks, the contact poses are adjusted along the tool direction of motion so that a tool is guaranteed to touch the manipulandum (e.g., when pushing,

an irregular object may require a slightly different relative position between the tool and manipulandum). With this method, we are able to generate collision-free contact poses.

4.2.4 Star 2: Task-General Object Substitution

Star 2 utilizes the tool use learned by Star 1 and calculates the appropriate contact poses by first finding the alignment between the source and substitute object. Using this alignment and the relevant taxonomic knowledge, the tool trajectory is adjusted accordingly. Star 2 requires the same manual inputs as the application in Star 1, for example the start and goal pose of the manipulanda, or more task-specific parameters, such as th desired scale and rotation of a drawn letter "R" as will be the case in Sec. 4.3.

Three-step Alignment Algorithm

For all tasks, except Infinite-Effects Tasks, contact poses are obtained by calculating the alignment between the source and substitute objects. The contact poses of Infinite-Effects Tasks depend on both the desired effects and the alignment. When the two tools are of the same type or share a generic form factor such as two different types of hammers, often considering the entire shape of both tools (i.e., their global features) produces the best results. In the case of tasks like pushing where no generic tool form-factor exists, utilizing features like the contact area (i.e., local features) of the source tool is necessary. Therefore, we designed a three-step alignment algorithm that produces alignments between a source and substitute objects using both global (step one) and local features (step two) and selects the most appropriate one (step three). Since we consider local features, object meshes need to be segmented prior to applying this algorithm. The application of the three-step alignment algorithm differs slightly for tools and manipulanda.

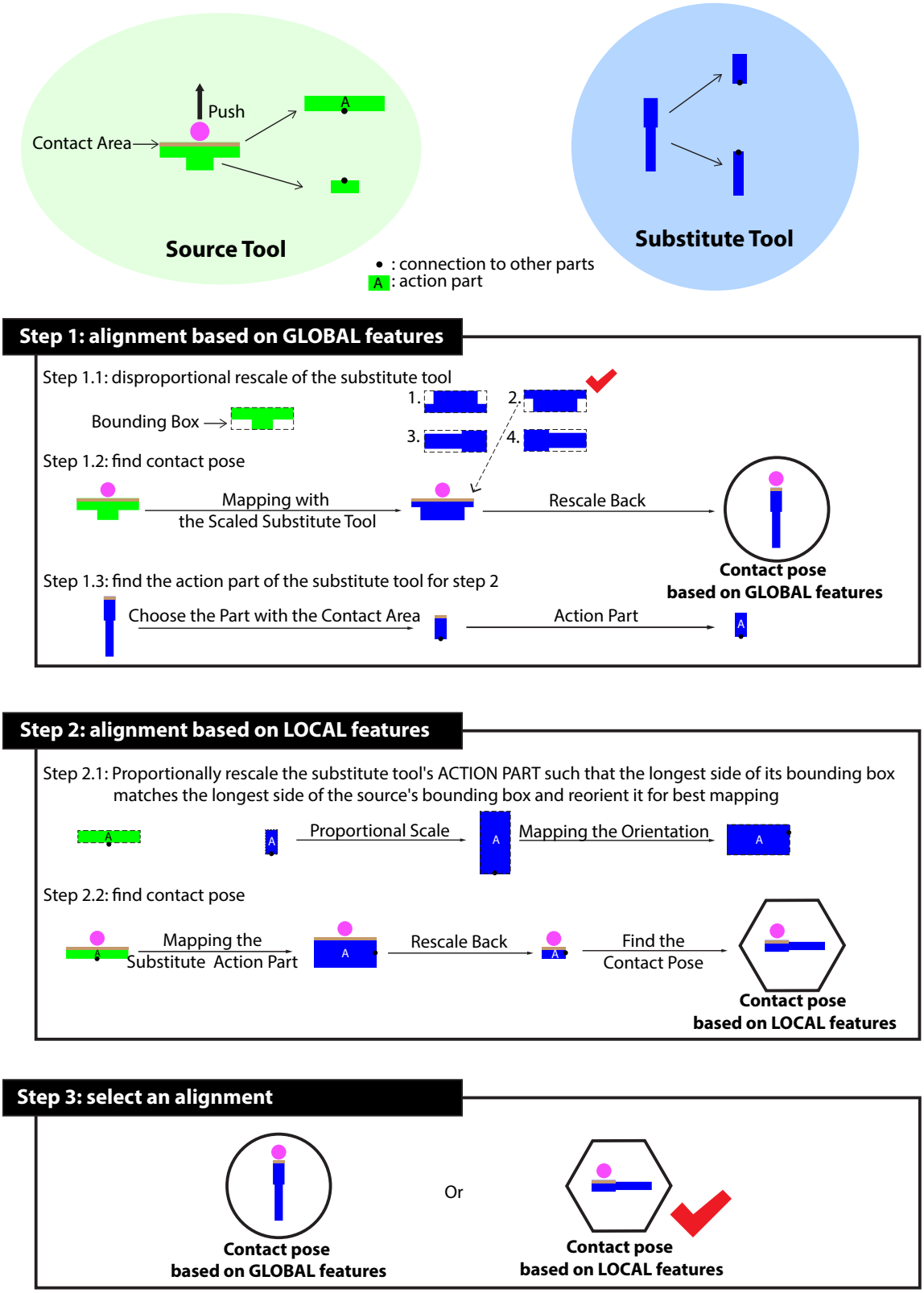


Figure 4.3: We present the alignment procedure for a hypothetical 2D tool substitution problem.

In order to segment a mesh, we utilized a method similar to a previous study (Abelha and Guerin, 2017) using the shape diameter function (CDF) with the CGAL library[6]. The number of clusters k were ranged from 2 to 8 with step 1, and the smoothness parameter λ ranged from 0.1 to 0.7 with step 0.1. Since no direct relation exists between the number of clusters k and the results of the segmentation, the number of clusters with the greatest number of results was chosen as k_{chosen}. Since, in most instances, the object with only one cluster is undesirable, k_{chosen} was allowed to be one only if the number of results with one cluster was significantly more than the number of clusters with the second greatest number of results. The segmentation was randomly chosen from all the segmentations with k_{chosen} clusters due to similarity.

Figure 4.3 depicts our process for finding contact poses given segmented tool models, by finding the alignment $T_{sub-tool}^{src-tool}$ between the source and substitute meshes. The goal of the first step is to find the alignment based on global geometric features. In this step, the substitute objects are rescaled disproportionally so that their bounding boxes share the same size as the bounding box of the source objects, and reoriented along the axes of the bounding box. As an object can be rescaled and reoriented in multiple ways, the one that is most similar to the source object is chosen as the alignment based on global features. The similarity is measured by the averaged minimum Euclidean distance between the points of the two point clouds when the centers of the two objects are aligned. The contact area on the substitute object is chosen by proximity to the contact area on the source object. The segment containing the contact area is chosen to be the action part which is used in step two. If the contact area is distributed across multiple segments, then the action part is chosen to be the contact area itself rather than any individual segment. As a result, we do not rely on the correctness of the segmentation. The goal of the second step is to find the alignment based on local geometric features. In order to find this

[6]https://doc.cgal.org/latest/Surface_mesh_segmentation/index.html

alignment and the corresponding contact area, the two action parts are mapped in a similar manner except that the substitute action part is rescaled proportionally, and the alignment of the two action parts uses modified iterative closest point (ICP) registration (Rusinkiewicz and Levoy, 2001). In step three, of the two contact areas found in the two steps, the candidate with the highest similarity score is chosen along with its corresponding contact pose and the alignment of the tools $T_{sub-tool}^{src-tool}$ is thus found.

The manipulanda do not need to be decomposed into action and grasping parts like tools do. Therefore, the contact area is assumed as the action part, and the algorithm to find the alignment poses of the substitute manipulanda $T_{sub-man}^{src-man}$ is otherwise the same as finding the alignment of the substitute tool. For Infinite-Effects Tasks, the alignment of the manipulanda is not needed since the geometric features of the manipulanda do not decide the alignment. Therefore, it is handled in the same manner as the source manipulanda in that the start pose is updated to incorporate the desired effect. The alignment, in this case, is set to be the identity matrix.

Generating Tool Trajectories

Given the alignment resulting from the three-step alignment algorithm, the trajectory of the substitute tool can be found using the learned source tool trajectory with adjustments based on the taxonomic knowledge if necessary (see Section 4.2.3). With the obtained tool trajectory, the end-effector trajectory is calculated from the tool trajectory in the same way as Star 1, except that the functional component is rescaled based on the size of the substitute manipulandum relative to the source for Non-Pose-Based tasks.

To find a candidate tool trajectory, an equivalent trajectory of the source tool acting upon an equivalent source manipulanda (i.e., the equivalent start pose and

goal pose of the manipulandum is calculated with $T^{world}_{start-sub-man} \times (T^{src-man}_{sub-man})^{-1}$ and $T^{world}_{end-sub-man} \times (T^{src-man}_{sub-man})^{-1}$, respectively) is first found. Then each pose of the trajectory $T^{src-man}_{src-tool}$ is updated with $(T^{src-man}_{sub-man})^{-1} \times T^{src-man}_{src-tool} \times T^{src-tool}_{sub-tool}$ which calculates the trajectory of the substitute tool in the substitute manipulandum frame. The trajectory is then converted to the world frame. For Non-Pose-Based Tasks, the functional component of the trajectory is rescaled based on the relative size of the longest dimension of the source and substitute manipulandum. Multiple candidate tool trajectories are found and each corresponding to a contact pose chosen in the same way as in Star 1. The final tool trajectory is chosen from the candidate tool trajectories in the same way as in Star 1.

4.2.5 Star 3: Tool Use Transfer to Other Robot Platforms

As tool-use skills learned by Star 1 are represented independently of robot configurations, no additional algorithms were needed in order to enable skill transfer to different platforms that could perform the given task. This was facilitated by the development of a perception system that obtains the 3D poses of the tools and manipulanda from RGB-D cameras, though, in principle, any method that can accurately perceive these poses can be used. With the learned tool use and the perceived grasping, we calculate the end-effector trajectories and control the robot by leveraging existing inverse kinematics and motion planning libraries. In order to simplify motion control across different robot platforms, we implemented a Robot Operating System node that uses the same interface to control all three robots. This interface can be easily extended to accommodate more platforms.

The same mechanisms of partially executing a trajectory or completely aborting it mentioned in Section 4.2.3 also apply when the platforms being transferred to cannot execute the generated actions. Moreover, learning a class of contact poses also helps with finding viable solutions on different platforms. For example, when required to

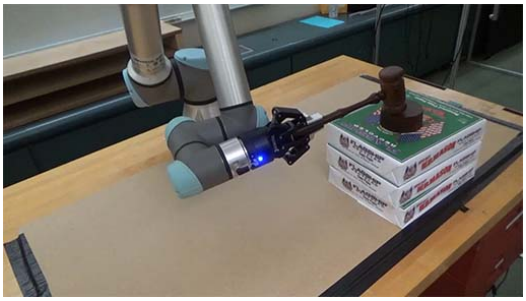
drive a nail with a hammer, a robot can choose to approach a manipulandum from any orientation, even those not appearing in the training set, which increases the viable kinematic solutions when a robot searches for motion planning.

4.3 Results

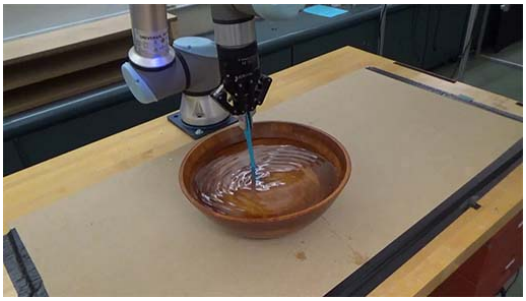
We tested Star 1 with seven tasks trained with minimal training samples via Learning from Demonstration (Argall et al., 2009). We tested Star 2 by providing three substitute tools and three manipulanda for each task. Finally, we conducted experiments for Star 3 that transferred the learned skills to two other robot platforms with different kinematic configurations.

4.3.1 Star 1: Learning and Applying Task-General Tool Use Skills

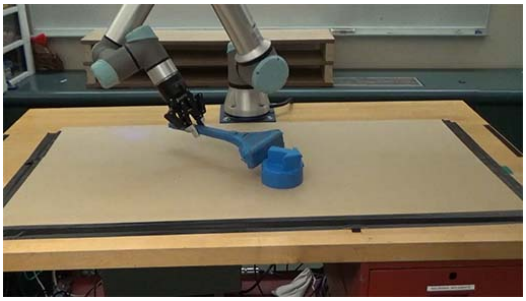
Figure 4.4 shows an example from each of the seven tasks with the source tools and manipulanda, and Figure 4.5a shows the testing environment. Six of the seven tasks were tested on a UR5e robot, and the screw-driving task was demonstrated on a simulated UR5e due to the higher perception accuracy required to align the tip of a screwdriver to the slot on the head of a screw. All tasks tested on the physical robot were evaluated quantitatively except for the writing task, which was included for demonstration purposes only. Creating quantitative metrics was sometimes challenging; while the pushing task could be evaluated with translation errors to the goal as had been done previously (Xie et al., 2019; Fitzgerald et al., 2019), other tasks were previously reported with only binary success or failure results (Pastor et al., 2009; Brandi et al., 2014) or success rates over multiple trials (Fang et al., 2020; Gajewski et al., 2019). When evaluating performance quantitatively, we used stricter methods (e.g., using loudness in decibels for the knocking task) when possible.



(a) Knocking.

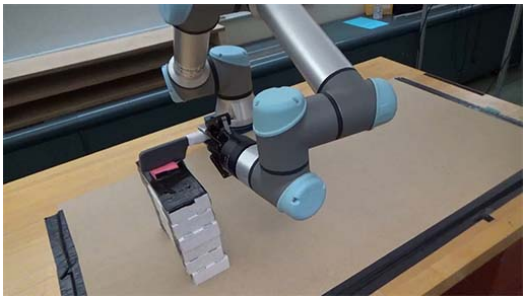


(b) Stirring.



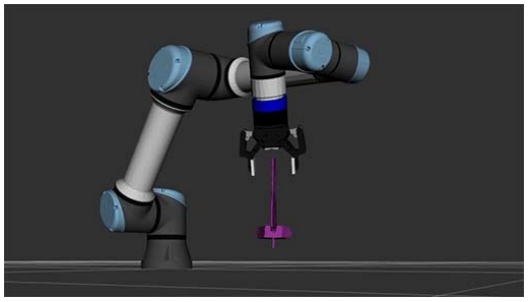
(c) Pushing.

(d) Scooping.



(e) Cutting.

(f) Writing.



(g) Screw-driving.

Figure 4.4: Our approach enables a robot to learn a wide variety of tool-use tasks. We demonstrate this using tasks such as (a) knocking, (b) stirring, (c) pushing, (d) scooping, (e) cutting, (f) writing, and (g) screw-driving.

(a) The workspace of the UR5e robot.

(b) The workspace of the Baxter robot.

(c) The workspace of the Kuka youBot robot.

Figure 4.5: Consistency across robot workspaces was maintained for the (a) UR5e, (b) Baxter, and (c) Kuka youBot platforms used in this work. This includes two Azure Kinect RGB-D placed on either side of the workspace.

Figure 4.6: TRI-STAR enables a robot to perform tool use skills irrespective of the tool's grasping pose. For each task, that is, knocking, stirring, pushing, scooping, and cutting, at least three different grasping poses were tested.

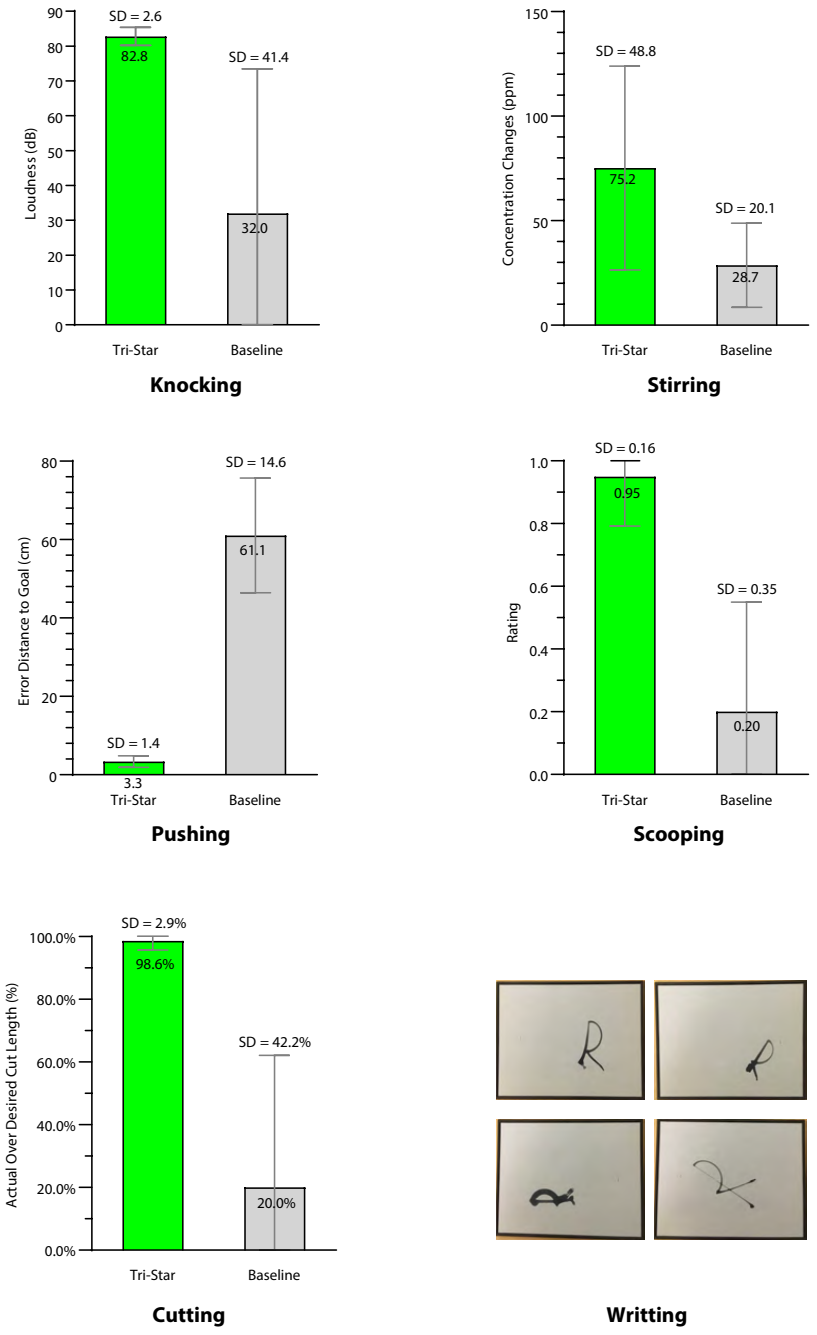


Figure 4.7: We compared Star 1 (green) performance against a baseline (gray) for knocking, stirring, pushing, scooping, and cutting using source tools and manipulanda. The pictures at the bottom right show the demonstrations of the writing task. The top left is an "R" using the same scale and rotation as the training sample. The top right, bottom left, and bottom right "R"s used the following scales and orientations: scale 1.0, orientation 270°; scale 0.8, orientation 30°; scale 1.5, orientation 300°.

The five tasks analyzed quantitatively were also compared with a baseline condition. We designed the baseline condition to accord with the common practice across task-general tool use learning frameworks of using the gripper pose as a proxy to the tool pose. Therefore, in the baseline condition, the robot repeated an end-effector trajectory in the task space of a training sample chosen randomly. For the five tasks, we tested ten trials per task per condition. Trials in which the robot was not able to follow the commanded trajectories were excluded. The start and goal poses of the manipulanda were altered in each trial. In both the experimental and baseline condition, the robot held tools with various poses as shown in Figure 4.6, a complexity that was not present in other studies. These poses were provided to the robot by the experimenters in order to impose pose variety (see Section 4.1.4 for motivation), though in principle TRI-STAR can accommodate autonomous grasping. Figure 4.7 summarizes the results. Details of testing each task are described below.

Knocking. The robot is required to strike an object with a hammer to produce sound. The robot successfully completed the task in 10 out of 10 trials in the testing condition, while its performance in the baseline condition was 4 out of 10 trials. We also measured the sound of each knock on the manipulandum using the Sound Meter app with a Samsung tablet placed close to the manipulandum. The average decibels, including the reading from unsuccessful trials, of the testing condition (mean (M) = 82.79 decibel (dB), Standard Deviation (SD) = 2.58 dB) was higher than the baseline condition (M = 32.00 dB, SD = 41.44 dB).

Stirring. The robot is required to stir the liquid with a spatula to dissolve salt. 0.25 tsp salt per liter was added to the room-temperature water and given several seconds to settle. The robot was allowed to stir for one minute or five circles, whichever lasted longer. Due to kinematic constraints, the grasps in the testing conditions were similar to the training pose. This constraint, along with the enforced grasping pose consistency across training and baseline conditions, resulted in both

training and testing conditions completing 10 of 10 trials. We also measured the concentration changes in part per million (ppm) before and after the stirring using a total dissolved solids meter. More salt dissolved in the testing condition ($M = 75.20$ ppm, $SD = 48.79$ ppm) than in the baseline condition ($M = 28.70$ ppm, $SD = 20.10$ ppm).

Pushing. The robot is required to push a blue item to the goal position chosen randomly by the experimenters. The manipulandum was pushed closer to the goal position in the testing condition (translation error: $M = 3.36$ centimeters (cm), $SD = 1.45$ cm) than in the baseline condition ($M = 61.06$ cm, $SD = 14.62$ cm). Our translation error in the testing condition is consistent with a recent study (Xie et al., 2019; $M = 6.37$ cm, $SD = 5.33$ cm) which also utilized perceptual data from raw sensor readings. The translation errors were mainly due to perception errors. This is supported by the significantly reduced translation error ($M = 0.013$ cm, $SD = 0.0074$ cm) observed when performing the same experiments using a simulated UR5e robot with perfect perception.

Scooping. The robot is required to scoop a rubber duck placed on top of packing peanuts. The performance was rated as 1 if the robot successfully scooped the manipulandum, 0.5 if the rubber duck slipped away but the robot scooped surrounding packing material, and 0 if the robot failed to scoop anything. The robot scooped the manipulandum more successfully in the testing condition ($M = 0.95$, $SD = 0.16$) than in the baseline condition ($M = 0.20$, $SD = 0.35$).

Cutting. The robot is required to cut a piece of putty in half. We measured the percentage length of the actual cut over the length of the intended cut. Even with a relaxed criterion accepting cuts as shallow as 1mm depth in the baseline condition, the robot cut the putty more thoroughly in the testing condition ($M = 98.62\%$, $SD = 2.91\%$) than in the baseline condition ($M = 20.00\%$, $SD = 42.16\%$).

Writing. The robot is required to write the letter "R" at the chosen location

with the required scale and orientation. The required scale and orientation may or may not be included in the training samples. Figure 4.7 shows various letters "R" that the robot wrote.

Screw-driving. The robot is required to drive a screw placed at random locations and orientations in simulation. The simulated robot was able to complete all iterations of the task successfully.

4.3.2 Star 2: Task-General Object Substitution

Five tasks (knocking, stirring, pushing, scooping, and cutting) were tested on a UR5e robot. Other than using substitute objects, the experiments and evaluations in Star 2 were the same as those performed in Star 1. For each task, three pairs of substitute objects were tested, and all objects were appropriate for the tasks. In the baseline condition, a random contact area and a contact pose were chosen for each of the substitute objects. The trajectories were generated using the same method as the testing condition. Figure 4.8 shows the source and substitute objects. Figure 4.9 shows the alignment result of each substitute object with the source object in each task. Figure 4.10 summarizes the results of the five tasks. Details of each task are described below.

Knocking. All three substitute tools successfully struck the substitute manipulanda in all trials in the testing condition, while the performance dropped significantly in the baseline condition (i.e., at most 1 out of 10 trials for each tool-manipulandum combination). In a previous study with a similar task (Fang et al., 2020), the highest success rate on nail-hammering was 86.7% of all the substitute tools with tens of thousands of training samples. In the testing condition, the average loudness in the testing condition ($M = 65.62$ dB, $SD = 3.35$ dB) was higher than that of the baseline condition ($M = 4.34$ dB, $SD = 16.50$ dB), while the loudness was not measured in the previous study.

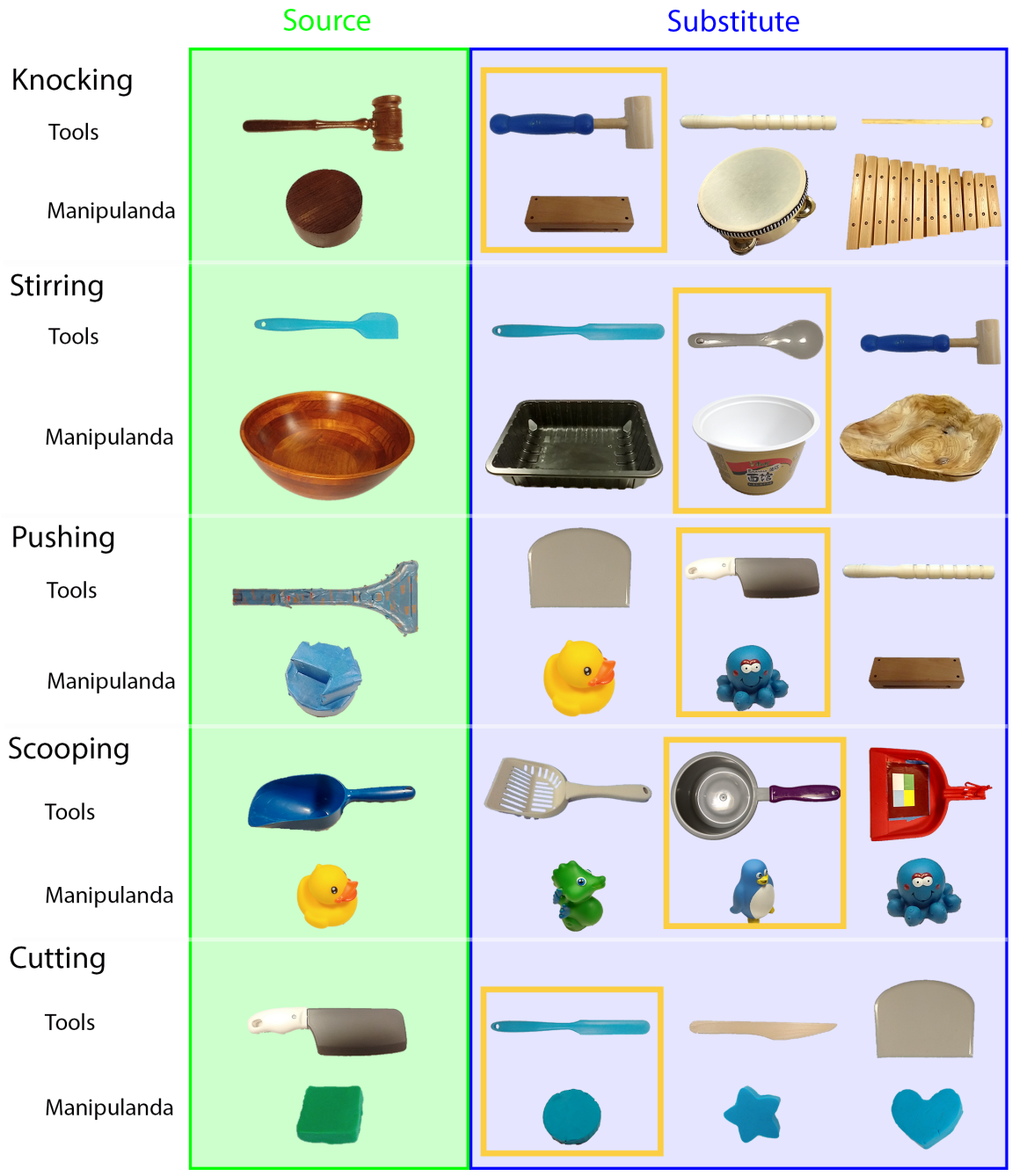


Figure 4.8: Star 2 enables a robot to transfer learned skills to novel tools and manipulanda. For each learned task, that is, knocking, stirring, pushing, scooping, and cutting, three substitute tools, and three substitute manipulanda were included in testing. The objects in the yellow frames were used as source objects in Star 3.

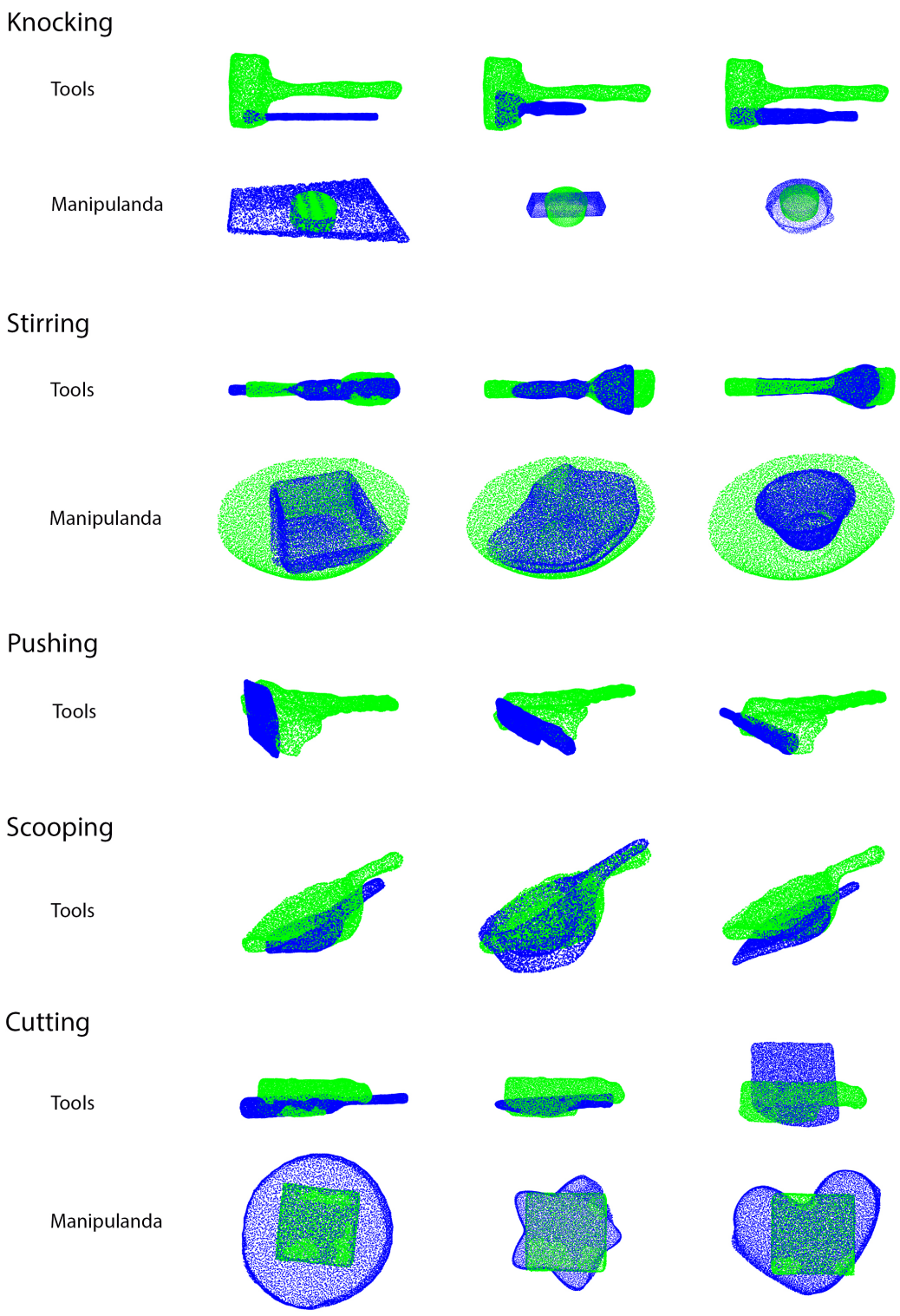


Figure 4.9: Here we show the results of aligning substitute objects to source objects (Star 2). The green point clouds are the source objects while the blue point clouds are the substitute objects. Manipulandum substitution for the pushing and scooping task is not geometry-dependent, but goal-dependent, and therefore, the alignment results are excluded in the figure.

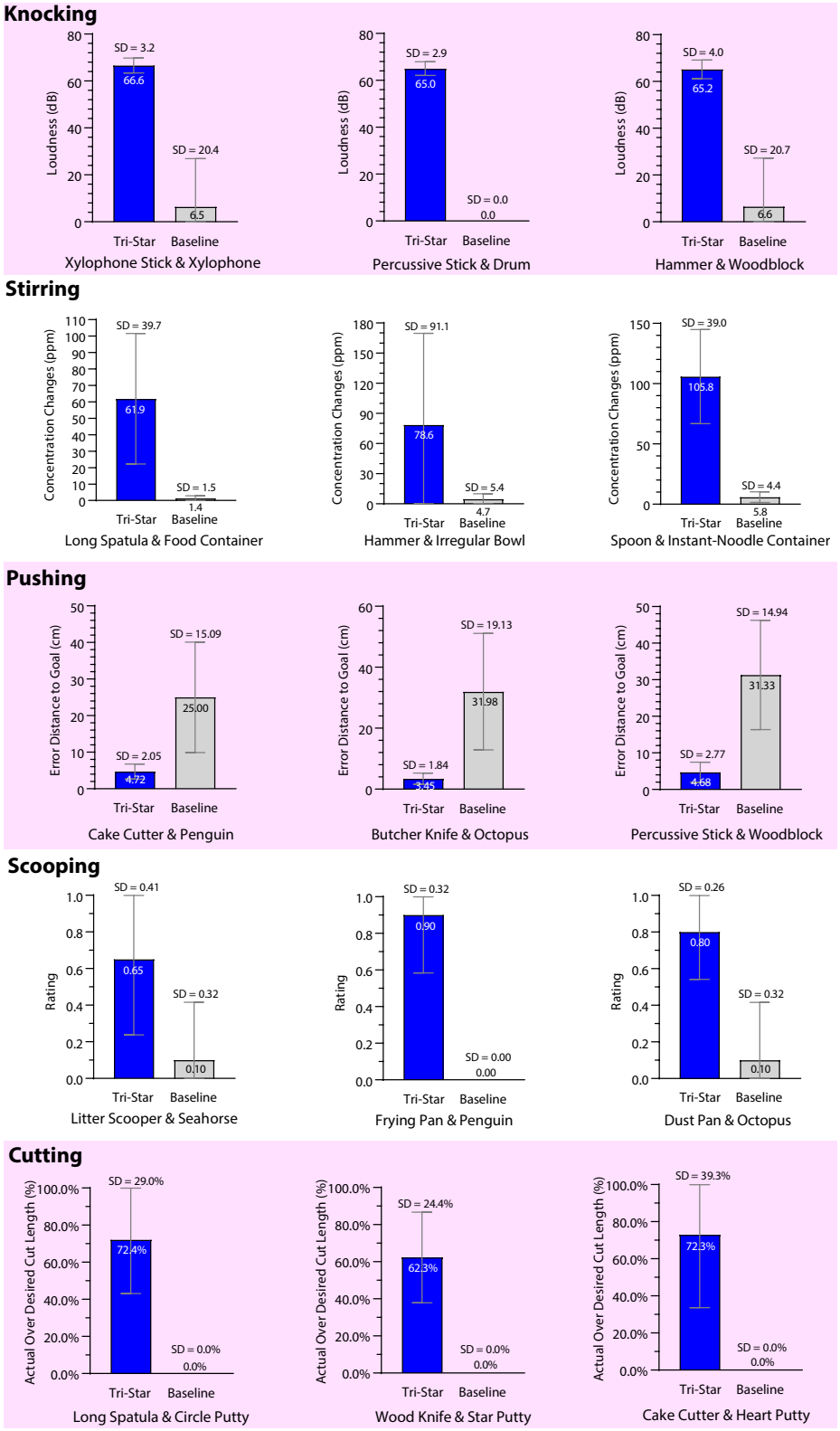


Figure 4.10: These graphs depict the results of tool substitution and manipulandum substitution (Star 2). The bar graphs show the results of using the substitute objects to perform knocking, stirring, pushing, scooping, and cutting. The bars compare Star 2's (blue) performance against the baseline (gray).

Stirring. All three substitute tools successfully stirred the room-temperature salted water in the substitute containers in all trials in the testing condition, while all substitute tools failed to stir in the baseline condition. More salt dissolved in the testing condition (concentration change: $M = 82.10$ ppm, $SD = 62.29$ ppm) than in the baseline condition ($M = 3.97$ ppm, $SD = 4.43$ ppm). We did not encounter another study that performed a similar task.

Pushing. The manipulanda were pushed closer to the goal in the testing condition (translation error: $M = 4.28$ cm, $SD = 2.26$ cm) than in the baseline condition ($M = 29.44$ cm, $SD = 16.24$ cm). In a previous study that also used raw sensor data to perceive the environment (Xie et al., 2019), the translation error using substitute tools and source manipulanda was similar ($M = 5.56$ cm, $SD = 4.13$ cm) to the current study but required more than 10^4 training samples.

Scooping. The substitute tools scooped the substitute manipulanda more successfully in the testing condition (rating: $M = 0.78$, $SD = 0.34$) than in the baseline condition ($M = 0.07$, $SD = 0.25$). In a previous study (Gajewski et al., 2019), the scooping task was tested only in simulation with substitute tools and source manipulanda, and no quantitative results (e.g, success rate) were provided.

Cutting. The robot cut the manipulanda more thoroughly in the testing condition (cut length percentage: $M = 78.33\%$, $SD = 33.95\%$) than in the baseline condition ($M = 6.67\%$, $SD = 25.37\%$) even with relaxed criteria in the baseline condition as mentioned in the Star 1 evaluation. In a previous study (Gajewski et al., 2019), the cutting task was tested only in simulation with substitute tools and source manipulanda, and no quantitative results (e.g., success rate) were provided.

4.3.3 Star 3: Skill Transfer To Other Robot Platforms

Six tasks (pushing, stirring, knocking, cutting, scooping, and writing) were used to test skill transfer from a UR5e robot to both a Baxter robot and a Kuka youBot

without additional training. Due to the size and payload limitations of Baxter and youBot, source tools different from Star 1 were chosen. The experiments were similar to the ones in Star 1. However, no baseline conditions were included in Star 3, and no comparisons were made with other studies since we did not encounter similar studies. Figure 4.5b and 4.5c show the testing environment of Baxter and youBot. The objects in the yellow frames of Figure 4.8 are the objects tested in Star 3. Star 3 only considered scenarios that the new platforms could complete if they were trained in the same way as the source platform. Therefore, the task configurations of all experiments were within the feasible workspace of the new robots. Figure 4.11 summarizes the results.

Knocking. All three robots successfully completed all trials. The loudness created by the UR5e (M = 75.43 dB, SD = 2.57 dB), Baxter (M = 74.04 dB, SD = 3.95 dB) and youBot (M = 73.89 dB, SD = 7.78 dB) were similar.

Stirring. All three robots successfully completed all trials. The concentration changes of the stirs by Baxter (M = 185.10 ppm, SD = 86.01 ppm) and youBot (M = 176.00 ppm, SD = 35.74 ppm) was slightly higher than the stirs by the UR5e (M = 160.60 ppm, SD = 43.71 ppm).

Pushing. YouBot (translation error: M = 2.40 cm, SD = 1.02cm) pushed the manipulanda slightly closer to the goal than UR5e (M = 3.78 cm, SD = 1.74 cm) or Baxter (M = 4.04 cm, SD = 2.25 cm), which was because of the shorter pushing length by youBot due to limited maximum reach compared with UR5e and Baxter. Since it is open-loop control, fewer errors are accumulated during this shortened course of pushing.

Scooping. UR5e (ratings: M = 0.90, SD = 0.21), Baxter (M = 0.90, SD = 0.21) and youBot (M = 0.85, SD = 0.24) performed equally well.

Cutting. The average cut length percentage cut of UR5e (M = 92.39%, SD = 7.75%) and youBot (M = 96.92%, SD = 3.44%) was slightly longer than Baxter

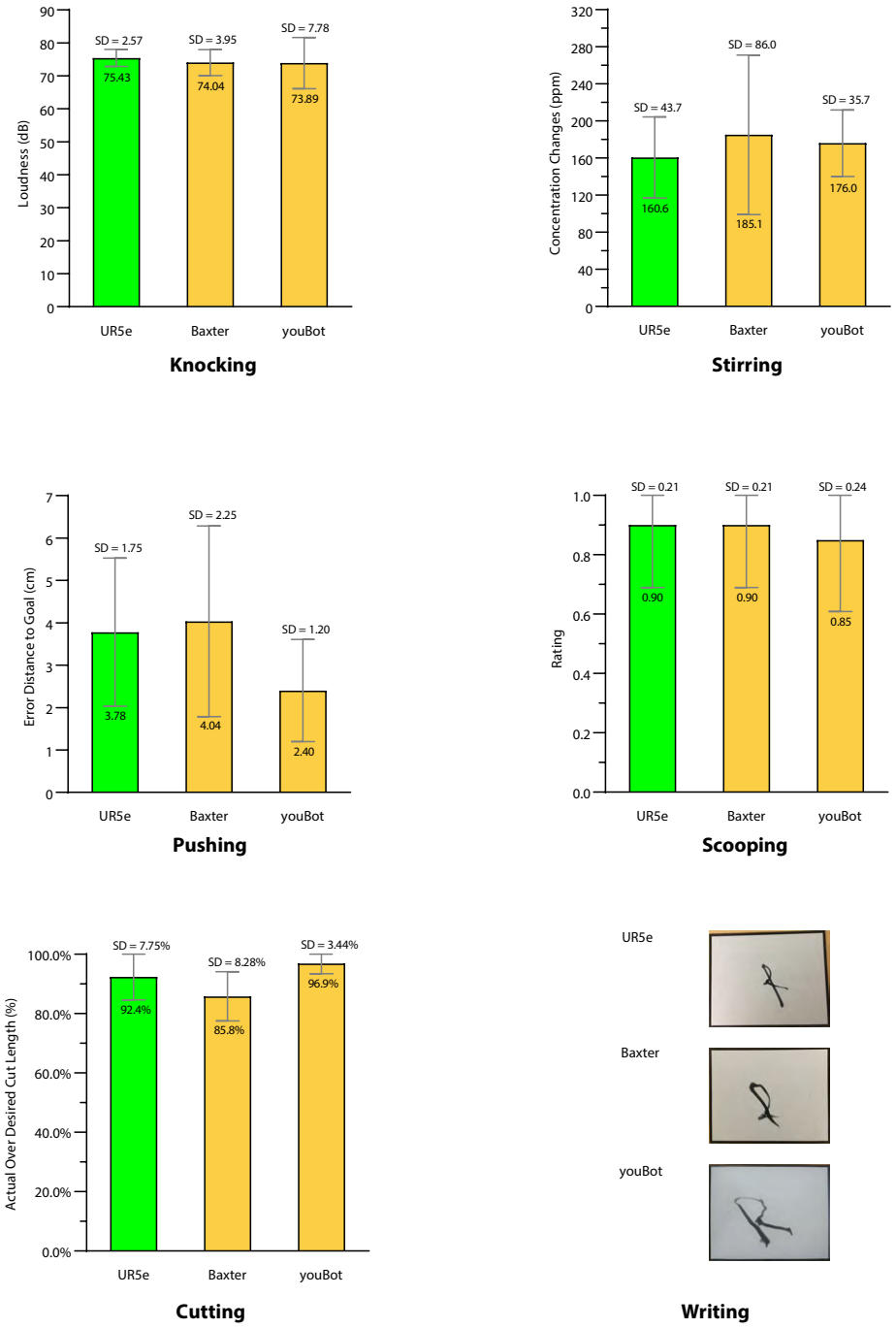


Figure 4.11: These graphs depict the results of tool use generalization across robot platforms (Star 3). The bar graphs include the results of the UR5e (green), Baxter (yellow), and youBot (yellow) using the source tool/manipulandum combinations for knocking, stirring, pushing, scooping, and cutting. The pictures at the bottom right demonstrate different robots writing "R" with trained scale and orientation.

($M = 85.83\%$, $SD = 8.28\%$), which was due to the difficulty in securing the spatula tightly in Baxter's gripper.

Writing. All three robots were able to repeat the letter "R. " Figure 4.8 shows the letter "R" with the trained scale and orientation written by the three robots.

Screw-driving. All three simulated robots completed the task successfully.

4.4 Discussion

The results show that the TRI-STAR framework learned a wide range of tasks, generalized the learned skills to substitute tools and manipulanda, and transferred the learned skills across robot platforms. This was achieved by using our task-oriented approach, which includes an affordance taxonomy and identified taxonomic knowledge which specifies knowledge shared across tasks that belong to particular task categories and minimizes the need for knowledge to be defined on a per-task basis. We center our discussion around the ways our framework improves upon the state-of-the-art in task-general tool use but also identify limitations of our approach.

4.4.1 Contribution 1: Task-Generality

TRI-STAR is a task-general tool use framework shown to learn, generalize, and transfer tool use for a variety of everyday tasks. Not all tool use algorithms are intended to be task-general as they assume pre-defined knowledge specific to individual tasks at either the learning or generalization stage. Three advances made it possible for TRI-STAR to be task-general. First, we summarized taxonomic knowledge of tasks which enables a multitude of tasks to be learned efficiently including potentially any undemonstrated tasks covered by one of the known taxonomic categories. Second, TRI-STAR can handle tasks with different contact pose requirements (e.g., pushing, knocking, and screw-driving) and different types of trajectories (e.g., circular peri-

odic trajectories including stirring, linear trajectories including cutting, non-linear trajectories including scooping, trajectories that could be either linear or non-linear including pushing, and complex trajectories with both linear and non-linear segments including writing) which allows a robot to work with a wide range of tasks. Third, TRI-STAR can generalize the tool use to tasks whose substitute tools and manipulanda may be geometrically-similar or geometrically-distinct objects since we made no assumptions about the shape of the objects, unlike previous approaches (Brandi et al., 2014; Fang et al., 2020). An added benefit of generalizing tool use to geometrically-distinct objects is that it can allow a robot to improvise the use of objects such that an object not designed for a task could be used when desired objects are unavailable.

4.4.2 Contribution 2: Data Efficiency

Task-general frameworks typically required a large training set size. However, training with a large sample size is time-consuming and thus impractical in time-sensitive domains like search-and-rescue. By leveraging taxonomic knowledge identified for each task category, TRI-STAR required only 20 examples to learn each task, and no additional training samples were needed by Star 2 to generalize the usage to substitute objects or by Star 3 to transfer the skills to other platforms. In contrast, previous studies required over 5,000 (Gajewski et al., 2019), 18,000 (Fang et al., 2020), and 20,000 (Xie et al., 2019) training samples. The small set of training samples needed for each task makes it time-efficient for TRI-STAR to learn new tasks and thus easy to be deployed as an application in the real world. Moreover, TRI-STAR experienced only a minor loss in performance while significantly reducing the necessary training samples.

Star 3 does not require additional training data nor extra algorithmic infrastructure to implement but rather updates a common representational schema (Cartesian trajectory) that is utilized in many tool use studies. To automate research work,

robots have been deployed in chemistry labs (Burger et al., 2020), where tasks and tools are standardized. The ability to transfer skills between robots could save researchers in each lab hundreds of hours of training time as skills could be shared across research labs. For robots in the factory or warehouse, it will be cost-efficient for skills to be transferred to new models without having to shut down the factory in order to debug compatibility-related problems. For other applications, platform-agnostic skill transfer would not merely be a convenience but could open entirely new applications. For example, for in-home robots, the prospect of training every single task by each individual is a nonstarter for most consumers, whereas having access to a shared library of skills may be more acceptable.

4.4.3 Limitations

While our results demonstrate the potential of our framework, it has limitations. First, the robot used position control only, rather than force control or feedback control, to learn and complete tasks. This potentially limits its effectiveness on tasks that require consideration of the forces being applied to the manipulanda such as nail-hammering, or tactile feedback such as inserting a key into a lock. Second, our framework only considers the geometric features of the tools and manipulanda and does not consider other properties (e.g., material, weight, texture), which may hinder the robot's ability to choose the most appropriate contact areas for tools like sandpaper that have a single abrasive surface but are otherwise geometrically uniform. Third, although our system calculated the grasping location on the tool, automatic grasping was not demonstrated in the evaluation.

Other limitations also exist for TRI-STAR. First, our framework assumes that all objects, including relevant objects in the environment, are rigid bodies with no joints (i.e., have 0 DoF). This assumption does not allow a robot to handle common tools such as scissors or washcloths or to perform tool use tasks on top of soft surfaces. Sec-

ond, our framework relies on accurate visual perception and structured environments, which is a common problem for non-marker-based perception systems and is an impediment to handling tasks that require highly accurate perception, such as surgery. Third, object alignment relies on full 3D models though ideally, this system should perform alignments using only partial point cloud data of both geometrically-similar and geometrically-distinct objects.

4.5 Summary

In this chapter, we presented TRI-STAR, a top-down hybrid framework for tool skill learning. We show that tailoring the subsymbolic skill-learning process using symbolic tool-affordance knowledge can improve the acquisition and utilization of tool skills, and facilitate the generalization of these skills to novel tools and manipulanda.

A top-down hybrid approach is well-suited for enabling these sorts of flexible tool-use capabilities. This stems from the fact that, for practical use cases, a tool-use robot may be expected to wield and manipulate unfamiliar variants of otherwise familiar objects. As we demonstrate here, recognizing and exploiting the invariant properties of tool-use tasks can prevent a robot from needlessly re-learning a skill that it already knows. This is only possible, however, if a robot is able to first reason about a tool use task at this higher level of analysis before attempting it, something readily enabled by a top-down approach.

Nevertheless, this work rests on the assumption that a robot already possesses a well-specified affordance taxonomy. While the taxonomy we developed here enables a robot to learn a wide variety of tool-use skills, it is not exhaustive. To make this approach truly practical for real-world use, it would need to be able to update and amend the affordance on its own, enabling it to learn and reason about entirely new classes of skills. As our affordance taxonomy is inherently causal, this would require

utilizing a bottom-up learning approach such as the one described in the previous chapter. Though we demonstrated our causal learning approach in a relatively simple environment consisting of relatively few causal variables, the work in this chapter suggests such simple models are sufficient for enabling sophisticated tool-use capabilities. Case in point, the affordance taxonomy developed here is predicated on the causal relationships between actions and manipulanda positions. It is precisely these sorts of relationships our approach is capable of learning, as demonstrated in Chapter 3. Our experiments in this chapter demonstrate that, in the context of tool use, even simple causal relationships can be exploited to great effect.

Beyond tool use, such an approach can also confer other HRC-relevant benefits as well. In the following chapter, we show how top-down hybrids can be used to improve the fluency and naturalness of HRCs by expanding a robot's ability to understand user commands.

Chapter 5

Top-down Hybrids: How Can Abstractions Benefit Natural Language Understanding?[1]

In the preceding chapter, we showed that structured task representations can be leveraged to improve the way a robot learns and reasons about its physical interactions with the environment. In this chapter, we shift our focus to social interactions, and apply some of these same insights to improving communication for human-robot collaborative (HRC) manufacturing tasks. In particular, we present a method for enabling a robot to utilize learned representations of joint tasks to contextualize user commands. This is accomplished by maintaining and incrementally updating separate "speech" and "context" models that jointly classify a collaborator's utterance. This enables a robot to interpret the sorts of highly semantically ambiguous utterances common to real-world collaborations, thus improving the naturalness of the HRC. We evaluate the efficacy of the system on a collaborative construction task

[1]Portions of this Chapter were originally published as: J. Brawer, O. Mangin, A. Roncone , S. Widder, B. Scassellati. Situated Human-Robot Collaboration: predicting intent from grounded natural language. In *Intelligent Robots and Systems (IROS)*, 2018. (Brawer et al., 2018)

with an autonomous robot and human participants. We first demonstrate that our system is capable of acquiring and deploying new task representations from limited and naturalistic data sets, and without any prior domain knowledge of language or the task itself. Finally, we show that our system is capable of significantly improving performance on an unfamiliar task after a one-shot exposure.

5.1 Introduction

The field of HRC is tasked with designing proactive and autonomous robot collaborators able to complement the superior capabilities of human workers to maximize throughput, improve safety of the workplace, and reduce cognitive load on humans. The general application domain for HRC is composed of a robot that collaborates with humans on a joint task such as furniture assembly (Knepper et al., 2013; Roncone et al., 2017), assembly lines (Johannsmeier and Haddadin, 2017), or other factory-related applications (Hayes and Scassellati, 2015; Tellex et al., 2014b). However, state of the art technologies still rely on sterile and rigid interactions that resort to turn-taking behaviors (Johannsmeier and Haddadin, 2017), tele-operation, or more generally limited autonomy and decision making capabilities (Bütepage and Kragic, 2017).

Conversely, human-human interaction (HHI) during teamwork does not show this friction. Fluent and natural HHIs are multimodal (Wahn et al., 2016), highly contextual and situated (Shah and Breazeal, 2010). This is particularly true when coordination during teamwork is attended through natural language. Humans resolve the natural ambiguities of speech by integrating verbal with non-verbal cues and, importantly, by grounding speech to the physical domain of the interaction—e.g., through implicature (Gazdar, 1980) or lexical entrainment (Iio et al., 2009; Lohse et al., 2008).

Yet, despite evidence of the importance of situated natural language in HHI,

achieving the same level of richness still represents a significant challenge for HRI in general and HRC in particular. Reasons for this are specific to HRC, e.g., the presence of noise in environments such as those commonly found in factories.

Noisy environments may result in failure to recognize significant portions of an utterance—if not the totality of it. This not only leads to erroneous naming of specific actions and objects, but also makes the structure of sentences harder to parse by natural language understanding (NLU) algorithms that exploit syntax. Most notably, impediments to deploying effective HRC interactions are also to be found in the very nature of the communication itself. Communication during collaboration often occurs in a time-constrained context, is highly goal-oriented, typically requires a high success rate in order to be effective, is domain-dependent, and often features mutual adaptation between peers. The time constraint during collaboration pressures agents to make shorter utterances that might not be well-formed sentences; the noise and the need for unambiguity favor some classes of words over others, often resulting in a highly domain-specific language. All these factors greatly hamper the deployment of standard NLU techniques to HRC. State-of-the-art technologies resort extensively to hand-coded domain knowledge, or require training on large datasets (e.g., Floridi and Chiriatti, 2020; Chowdhery et al., 2022)—most of which are taken from descriptive text and are borrowed from different contexts that do not necessarily leverage the specific domain knowledge. Still, to achieve the level of fluency seen in HHIs, a core ability of future generations of robots will be for them to collaborate with humans through the situated interactions with which humans are most comfortable (Breazeal et al., 2004).

Robots, in order to become proficient collaborators, should be able to exploit *context* in order to ground ambiguous and referential speech. However this is a more complex task than straightforwardly deploying NLU algorithms to HRC scenarios. Effective collaboration requires fast adaptation to different tasks and/or user prefer-

ences, a feature for which systems trained on corpora composed of billions of sentences do not allow. Further, a different kind of context awareness is needed. An HRC scenario constrains the verbal interaction to a very specific physical environment, and this affords the unique opportunity to latch a narrower context to the bigger NLP problem, which becomes then more tractable.

In this work, we implement a situated HRC system that integrates verbal instructions from a human partner with contextual information in the form of a task model. The proposed system learns task representations from demonstration, without requiring hand-coded domain linguistic or a priori task knowledge, on a task that is designed to trigger ambiguous, referential speech due to the use of parts and tools that are challenging to refer to verbally. Our experiments demonstrate that the system dynamically leverages linguistic and contextual information to provide support to the human worker; furthermore, it is capable of accomplishing this given a minimal set of noisy and naturalistic data. Additionally, the system is capable of learning online effective representations of tasks from one shot exposures, over real collaborative interactions.

5.2 Background and Related Work

Although communication is pervasive in collaborative activities, most HRC systems are not yet capable of handling the variability of natural communication. On the other hand, the large research in NLP and NLU mostly focuses on corpora of written language, that, by nature, differ from the short-term, context bounded, goal-oriented typical utterances that arise during collaborative activities. Indeed, collaborative scenarios typically include resource constraints that change the nature of the communication strategies by, for example, favoring explicit or implicit references (e.g., time pressure, Shah and Breazeal, 2010). Overall, communication serves various roles dur-

ing collaborative activities, such as sharing and aligning mental states (Briggs and Scheutz, 2011), providing confirmation (Sattar and Dudek, 2011), assigning roles and allocating subtasks (St Clair and Mataric, 2015; Roncone et al., 2017), or asking for help (Tellex et al., 2014a). However, due to the limitations of current technology, it is often necessary to develop strategies for the robot to handle misunderstanding: a robot may for example provide feedback on instructions that triggers users to adapt their speech and gestures (Lohse et al., 2008), or rely on more elaborate dialog templates (Deits et al., 2013).

Most research on natural language processing relies on pre-coded domain knowledge, or requires large datasets; in addition, it typically focuses on specific tasks like classification or translation, that are studied in isolation from real-world interactions with humans (Hirschberg and Manning, 2015). In robotics and human-computer interaction, instead, the interaction with users is paramount and data collection is expensive; for this reason, past works have augmented the amount of information available to such systems by *integrating language with context*. One example is seen in *multimodal fusion* approaches, that demonstrate how visual and acoustic information improve the understanding of commands from a human (Fransen et al., 2007). *Compositional instructions* also constitute a powerful knowledge representation to ground natural language commands: for example, Wang and colleagues demonstrate how an autonomous agent can learn to ground an unknown language in a simulated block assembly task from demonstrations (Wang et al., 2016b). Of great interest to this work is the field of *pragmatic modeling*, which introduces a model of the speaker's intentions to improve interpretation of goal-oriented utterances (Tellex et al., 2014b,a; Wang et al., 2016b). In this work, we explore a similar problem to these but in the context of a realistic HRC, where the language component is acquired from the human peer and not from typed text or generated by the robot.

More recently, a increasing body of work has focused on the application of NLP

approaches to HRC specifically. For instance, Cantrell and colleagues demonstrate how a robot can rely on dialog systems to acquire knowledge about new actions from a human (Cantrell et al., 2011). In addition, several studies targeted specific linguistic contents that are typical of collaborative environments. In such scenarios, humans often trigger references to spatial relationships and several methods have been developed to ground language on such constructs (Guadarrama et al., 2013; Kollar et al., 2010; Hemachandra et al., 2014; Tellex et al., 2011). Planning constraints also arise naturally as a way to provide instructions to collaborative robots (Howard et al., 2014; Cantrell et al., 2012), as well as reward functions (MacGlashan et al., 2015). In this work, we present a framework for the robot to learn from demonstrations how to respond to natural language commands. Our system takes advantage of contextual information, but importantly it does not assume previous knowledge about the language, the task, or the grammatical forms used.

5.3 Material and Methods

5.3.1 Experimental Setup

This chapter introduces an experiment designed around a human participant and a Baxter collaborative robot engaged in a construction task—more specifically a small-scale chair, developed in prior work (Zeylikman et al., 2018) and shown in Figure 5.1. The chair, depicted in Figure 5.2, requires nineteen individual parts to be built: seven dowels that act as legs and supports for the back, a chair seat, a chair back and ten connecting joints that fasten parts together. A single screwdriver is the only tool needed to secure a total of twelve screws.

For the purposes of this work, we confine the interaction to a master–slave configuration, where the robot is expected to provide support to the human upon request—similarly to Mangin et al. (2017) and Hayes and Scassellati (2015). More specifically,

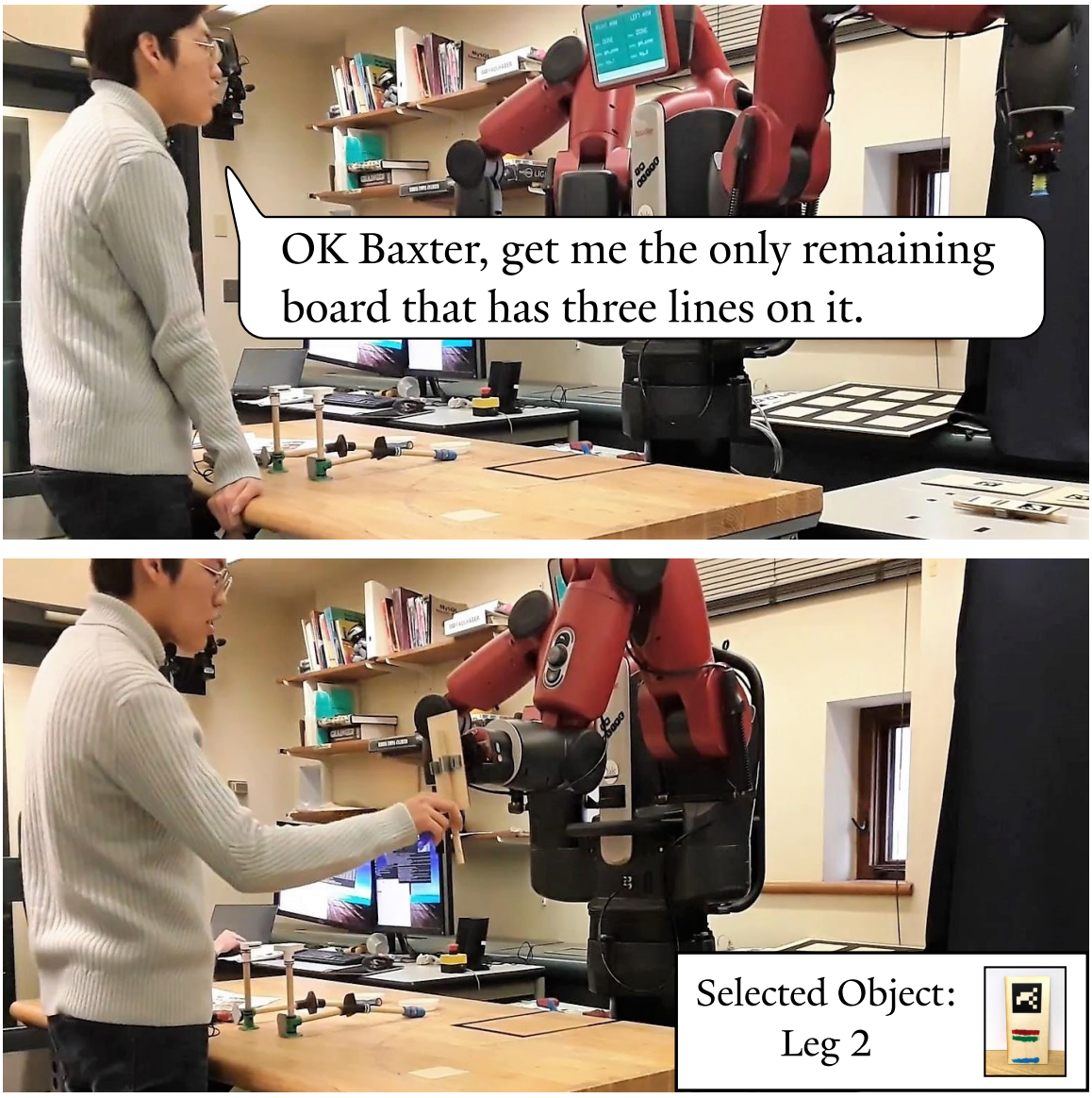


Figure 5.1: Naturalistic commands issued during a human-robot collaboration can be semantically ambiguous. Depicted here is an example of a user requesting a manufacturing component without a clear referent. Our proposed system integrates speech and symbolical contextual information to autonomously select the optimal part.

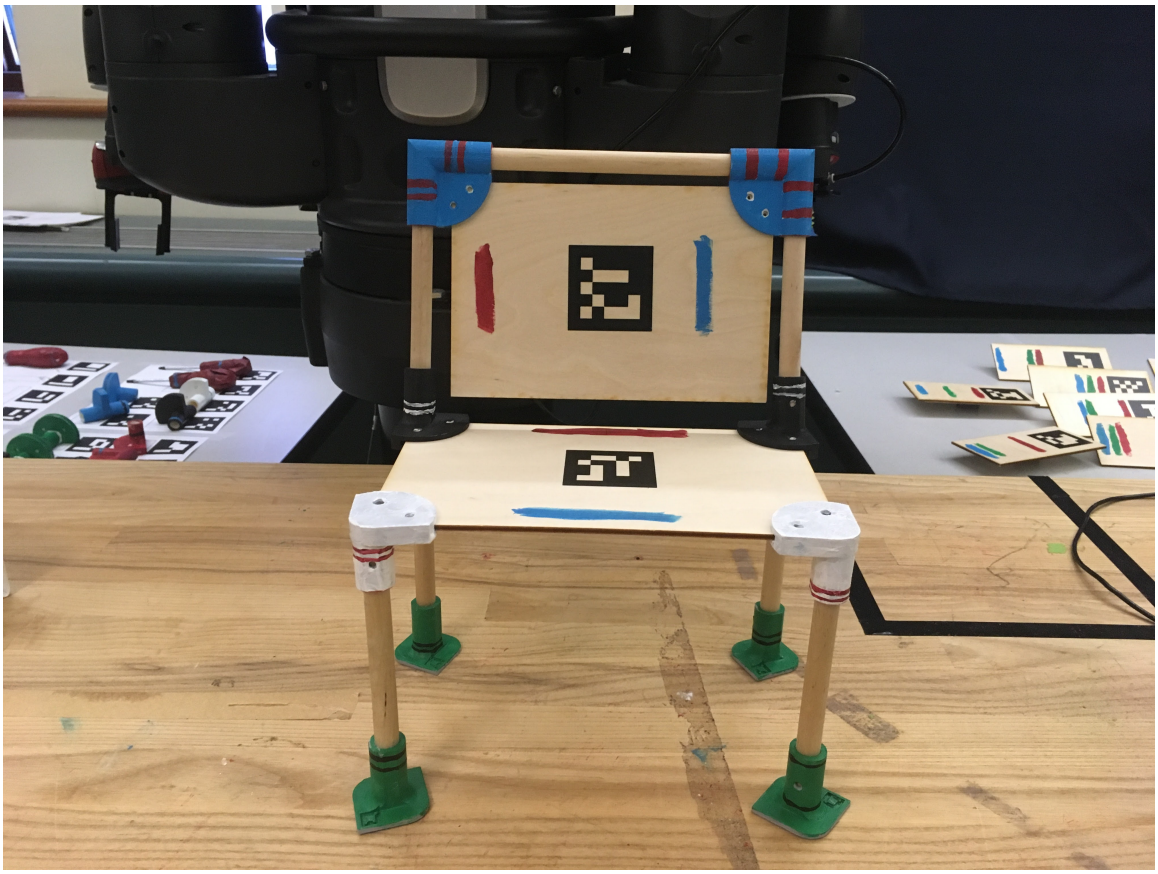


Figure 5.2: The application domain the human and the robot are tasked with is the joint construction of a small-scale chair, depicted above. Colored patterns were added to the chair components to increase visual noise and thus elicit potentially ambiguous commands from users.

Figure 5.3: Color patterns can be used to refer to objects and tools, but not unequivocally. For example, the two white pieces in the foreground differ only in the position of their red stripes. Similarly, using purely spatial relationships to refer to objects is difficult due to the large number of objects present.

the parts constituting the model set are placed in two pools of objects (at the right and left side of the robot, see Figure 5.1) that can be accessed by the robot exclusively. To successfully perform the task, the participant must ask for constituent parts and tools through speech commands. To facilitate the speech recognition system, we enrolled native English speakers exclusively. Crucially, participants were free to verbally interact with the robot *in any way they preferred*, in order to train the system with as natural an interaction as possible. Additionally, a number of 'dummy' objects were integrated into the object pool to add visual noise and increase ambiguity. To this end, another important design decision was to artificially increase the visual complexity of tools and pieces required for the assembly task (Figure 5.3). All of the pieces were 3-D printed, novel designs, and lacked clear or otherwise recognizable labels. These pieces were painted with distinct, but overlapping patterns to make verbally labeling these components more complex. Creating a sufficiently ambiguous task was

a crucial feature of this experiment, as a way to reproduce realistic environments in which a simple speech model is not sufficient for the robot's needs (see Sec. 5.3.2).

The Baxter robot is provided with a set of basic capabilities, encapsulated into a library of high-level actions originally developed in Roncone et al. (2017)[2]. The perception system is provided by ARuco (Garrido-Jurado et al., 2014), a library for generation and detection of fiducial markers. Each object in the workspace is provided with an unique marker, which is detected by the end effectors' cameras and then mapped into the robot's 3D operational space via the robot kinematics. In this work, the robot is simply tasked with picking up objects and tools and passing them to its human partner. Additionally, we employ the following software layers: i) a *web interface* to remotely receive feedback from and tele-operate the robot during collection of the training dataset; ii) a *speech-to-text (STT)* software that employs the state-of-the-art Google Cloud STT API to convert verbal commands into text strings; iii) a *feedback channel* visualized on the robot display to provide feedback to the participant about the robot's internal state (see Figure 5.1).

5.3.2 Data and Learning Algorithm

We propose a flexible approach for incrementally deriving a model of the desired robot's behavior from utterances and context. This is accomplished by maintaining and updating distinct models of the *speech* and the *context* in relation to robot actions, the outputs of which are combined for action selection. For a given utterance and the associated context, each model yields a probability distribution over actions. Our system assumes the independence of the utterance and context observations, given the desired action. For a uniform prior on robot actions, this results in the predicted probability of actions being proportional to the product of the predictions of the speech and utterance models. More formally, for an observed context $c_t \in C$

[2]https://github.com/ScazLab/human_robot_collaboration

and an utterance $u_t \in U$, the action $\hat{a}_t$ chosen by the system at time-step t is the action that maximizes:

$$\hat{a}_t = \arg\max_{a \in A_t} \left[p_{\text{speech}}(a|u_t) \cdot p_{\text{context}}(a|c_t) \right]$$

where A_t is the set of feasible actions at time t, which typically excludes objects that are absent from the picking area at a given time.

Here *context* refers to both the presence of objects in the workspace but also the action history of the robot—the sequence of successful actions taken up until the current moment. For the purpose of this experiment we introduce a simple context model that records counts of actions taken in any context. We consider contexts represented as action histories: $C = \cup_{m \in \mathbb{N}} A^m$ where A^m is the set of all sequences of m actions. Although the state of all possible contexts is intractable, the system only models here the distribution on actions from observed contexts and assumes a uniform probability on unknown contexts. Because contexts are sequences of successful actions—each sub-sequence prefixing an observed context is itself an observed context—we implement the model by storing the counts of successful occurrences of actions in the tree of observed contexts. For a given, known, context c and $N_{a,c}$ the count of occurrences of action a in context c, the model returns a probability:

$$p_{\text{context}}(a|c) = (1 - \varepsilon_{\text{context}}) \frac{N_{a,c}}{\sum_{a' \in A} N_{a',c}} + \frac{\varepsilon_{\text{context}}}{|A|}$$

where $\varepsilon_{\text{context}}$ is an exploration parameter (0.15 in the experiment) that accounts for new unexpected actions in known contexts.

For the purposes of this experiment, we use a speech model based on logistic regression to represent $\Pr(a|u_t)$. More precisely, each utterance command u_t is converted by the speech-to-text system and then represented as a bag of n-grams (i.e. contiguous sequences of n words in the utterance) of size one and two. The model

for this experiment is based on Pedregosa et al. (2011) for both the classifier and the feature extraction. For a given utterance u, represented as a vector x of n-grams counts, the logistic regression model includes a parameter vector θ_a for each action and returns an estimate of the action probability:

$$p_{\text{logistic}}(a|u) = \frac{C}{1 + e^{-\theta_a^T x}}$$

where C is a normalization constant. We then compute:

$$p_{\text{speech}}(a|u) = (1 - \varepsilon_{\text{context}}) \cdot p_{\text{logistic}}(a|u) + \frac{\varepsilon_{\text{context}}}{|A|}$$

in which $\varepsilon_{\text{context}}$ accounts for the occurrence of irrelevant speech commands (0.15 in the experiment).

5.3.3 Data Collection Phase

We collected data from recordings of human interactions with a teleoperated robot. All training data was recorded from the chair building task, that each participant built three times, according to three different orderings (denoted as *instructions* A, B, and C in Table 5.1), that were provided through instruction sheets. Each instruction sheet simply includes a list of steps that the participant had to follow. Each step is represented by a picture of the part to ask the robot for and a picture of the current state of the chair being built. We rotated the order in which each participant was asked to build the chair, alternating between ABC, BCA, and CAB.

After receiving the instructions for the task, each participant was familiarized to the process of receiving parts from the robot, which includes pressing a button to trigger release of the part. During the explanation, the experimenters avoided the use of any other word than "part" to refer to the various elements of the assembly, in order not to bias the vocabulary later used by the participants. The instructions

Table 5.1: Three instruction sets (**A**, **B**, and **C**), and their corresponding variants (**A'**, **B'**, and **C'**) were used during data collection and experimental trials. Bolded components indicate steps that differ across instruction set variants.

A / A'	**B / B'**	**C / C'**
foot_1	foot_1	foot_1
leg_1	foot_2	leg_3
screwdriver	foot_3	screwdriver
foot_2	foot_4	back_1
leg_2	leg_1	leg_5
foot_3	screwdriver	top_2
leg_3	leg_2	foot_2
foot_4	leg_3	leg_4
leg_4	leg_4	back_2
front_1	front_1	leg_6
front_3	front_3	top_1
back_1	back_1	leg_7
back_2	back_2	back_2
seat / **leg_5**	seat / **leg_5**	foot_3 / **seat**
leg_5 / **leg_6**	leg_5 / **leg_6**	leg_1 / **foot_4**
leg_6 / **top_1**	leg_6 / **top_1**	front_1 / **leg_1**
top_2 / **top_2**	top_2 / **top_2**	foot_4 / **front_3**
top_1 / **leg_7**	top_1 / **leg_7**	leg_2 / **foot_3**
leg_7 / **back**	leg_7 / **back**	front_3 / **leg_2**
back / **seat**	back / **seat**	seat / **front_1**

specified that participants needed to refer unambiguously to the part they wanted from the robot.

During each of the three assemblies, an experimenter was waiting for the participant to formulate an unambiguous request and was then triggering the actions from the robot. However, the participants were led to believe that the robot was operating autonomously in order to elicit the most naturalistic utterances possible. All transcribed utterances (and corresponding robot actions) were collected and used later on to train the system. Importantly, *all* the sentences were collected, without filtering out bad utterances and/or broken requests. In total, 626 pairs of requested objects and actions were collected from 12 participants.

5.3.4 Evaluation

In order to evaluate the efficacy of the joint context and speech model system, we deployed the trained model on the robot and conducted autonomous construction trials with 11 participants. As during data collection, the robot supported the construction of three chairs by retrieving parts based on speech commands issued by participants. In addition to tasks A, B, and C we introduced three corresponding permutations: A', B', C' (here referred to as *prime* tasks). For the prime tasks, only the latter third of the instructions were permuted allowing for the performance of the model to be evaluated in a divergent and unfamiliar context. Participants were provided one of three sets of instructions: $AC'C'$, $BA'A'$, or $CB'B'$. The prime tasks were repeated by each participant in order to gauge how effectively the system was able to learn from the previous trial.

In accordance with the data collection phase, participants constructed the chair in the order specified by the instructions sets. In the event that an incorrect piece was retrieved by the robot, the participants were instructed to press a red button on the robot's end effector resulting in the robot returning the piece to its original location.

This was repeated until the robot retrieved the correct piece. The performance of the system was evaluated based on the number of occurrences of such button presses during each trial.

5.4 Results

5.4.1 Collected Data

In this experiment we initially trained a decision process as a classifier on the 20 actions from a little more than 600 samples collected from 12 participants over 3 assemblies per participant. Additionally, further training was acquired online during experimental sessions, with learning persisting across the three trials. Thus we demonstrate the feasibility of training a supportive robot on a limited amount of data, from a relatively unconstrained scenario, and without any assumption on the language used by the participants, outside of the availability of a speech to text system. This setting contrasts sharply with the large amount of data required by typical NLP systems. In particular it brings training to a range feasible for real-world HRC and robotics scenario, where data collection is both hard to constrain and costly.

Table 5.2 shows a selection of utterances for one action, chosen randomly from our training set. The table displays the variability of the utterances used to request the same object, as well as the transcription errors from the speech to text system. In particular, several utterances are not correctly transcribed, some are truncated, and some are not associated with the correct object. They illustrate the difficulty of acquiring clean training data from real world scenarios, even when using a teleoperated robot.

Table 5.2: Ten utterances from the randomly chosen 'foot_4' object, as detected by the speech-to-text system.

	Utterance
1	"Green Leaf shaped green leaf shaped object on your right arm with black pants"
2	"I had the last 3 months and then we're one black lines at the top and wants to bottom"
3	"the leaf shaped object green colored with One Bank on black bun on top and one on the bottom"
4	"thank you can I have another green part that has one black line on the top one black line on the bottom on your right arm"
5	"Baxter can you hand me"
6	"I need the greens Green Park and it has a hole in between the black to Circus"
7	"I want the green part with one black stripe at the top and one black stripe at the bottom"
8	"vodka the Green Bay's peace"
9	"give me the green part with black line on the top and the bottom"
10	"underneath the rectangular blue line on the top and in the middle Back Square private line"

5.4.2 User Interaction

Trial 1 utilized one of the three instructions from the training phase (either A, B, or C). Trials 2 and 3 had the participants repeating one of the novel *prime* permutations (either A', B', or C'). The repetition of prime tasks allowed us to evaluate how rapidly and effectively the system was able to learn new tasks.

Figure 5.4 presents the classification median error rate of the algorithm for the utterances captured from 11 participants across the three trials. No errors were observed in the familiar tasks trial, suggesting that the system robustly learned the structure of the three tasks comprising the trial. While the model performed noticeably worse on the unfamiliar task trial, a paired t-test revealed it's acquired familiarity with the task boosted performance significantly ($p < 0.001$) on the second attempt. In Figure 5.5 the effects of learning are made more apparent. Across the unfamiliar tasks

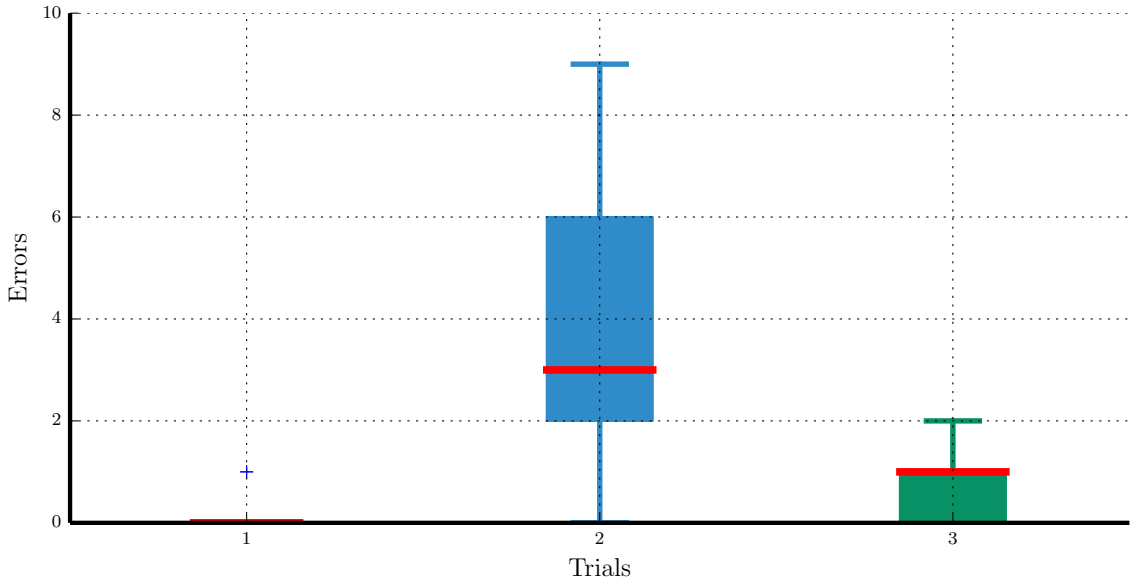


Figure 5.4: We present the errors per trial across participants. Here an error is an incorrect action taken by the robot. A paired t-test revealed a significant decrease in error rate across trials 2 and 3 ($p < 0.001$). This suggests that the system is capable of effectively learning new tasks from limited data.

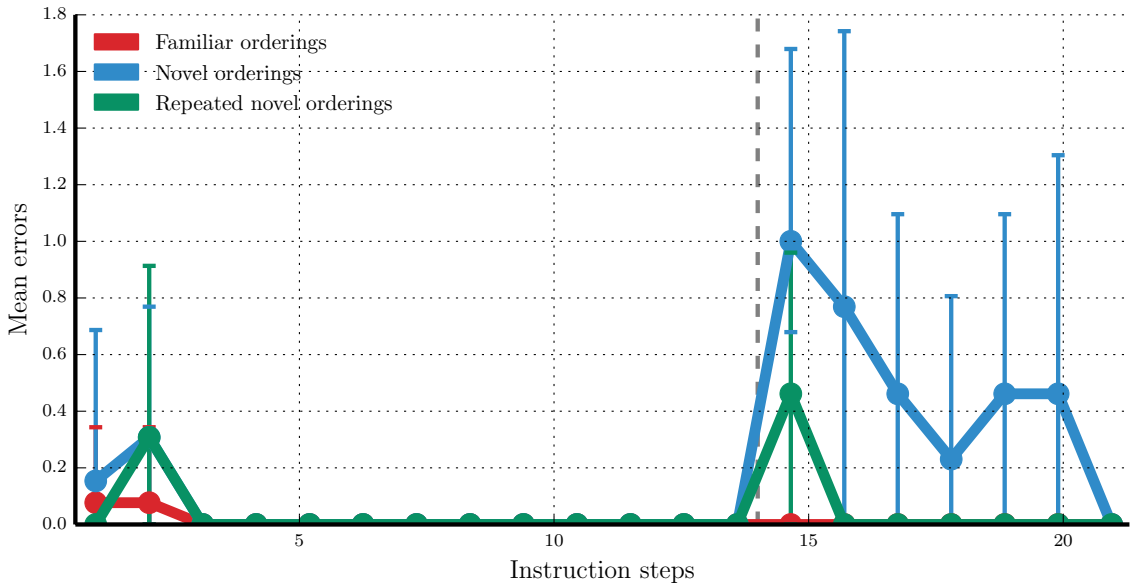


Figure 5.5: We present errors across instruction steps. The dashed black line indicates the step at which the instructions diverge from those used in the training set for the trails denoted by the blue and green lines.

Table 5.3: We present the average number of errors made by the robot (in parenthesis, standard deviation) per experimental trial for each model.

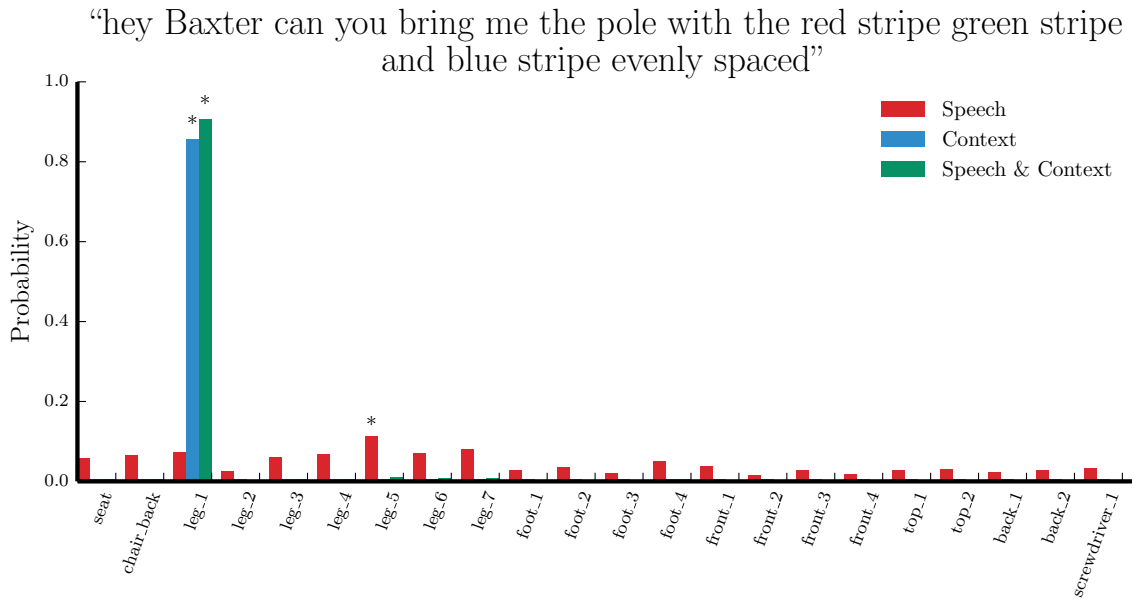
	Speech	Context	Speech & Context
Familiar	10.1 (2.16)	0.6 (0.48)	**0.0** (0.0)
Unfamiliar	10.8 (2.60)	6.7 (0.64)	**3.6** (1.11)
Repeated unfamiliar	10.3 (2.60)	1.7 (0.64)	**0.9** (0.53)

trial and the repetition trial, performance is worst at instruction step 14—the step at which the prime instructions diverge from their corresponding originals. However, on the second attempt, there are comparatively fewer errors at step 14, and indeed no errors for subsequent steps, supporting the models ability to rapidly acquire task structures from limited data.

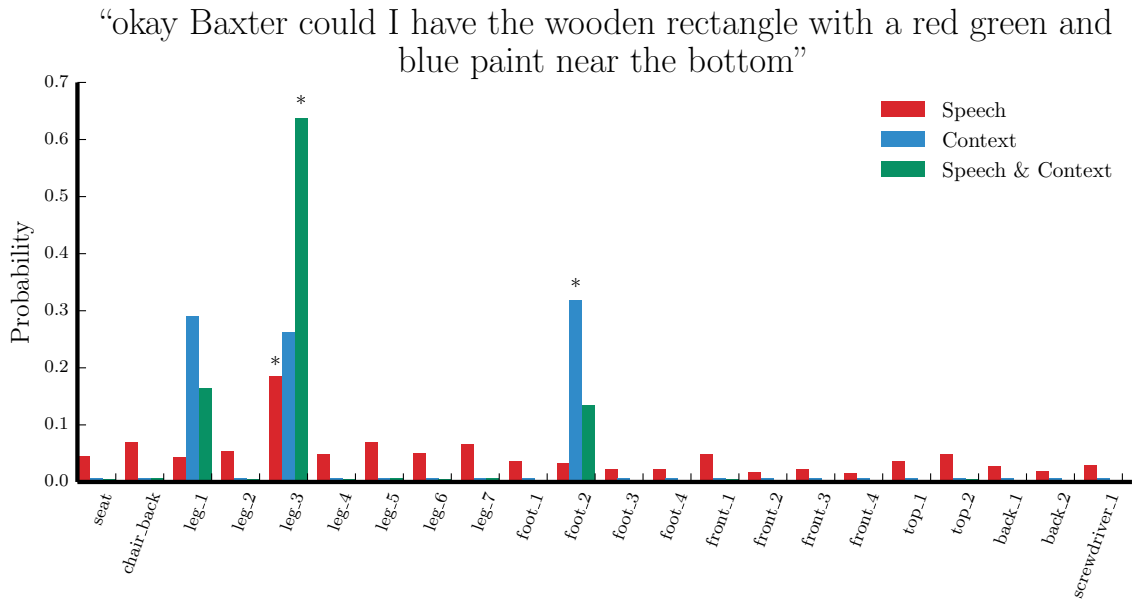
In addition, we compared the error rate of the speech, context, and joint speech and context models in simulation using the data collected from the experimental trials. In order to compare the three models on the data where only one was running, the error rate is the number of robot errors on first attempt. In other words, repetition of the command by the participant after an incorrect action from the robot are discarded. As reported in Table 5.3, the joint speech and context model produced the fewest errors on all trials, supporting the efficacy of this approach.

5.5 Discussion

We present a flexible system capable of building incremental models of speech and context information for producing desired behaviors. In addition, we present results from a baseline HRC experiment that had a participant and an autonomous robot complete an assembly task. For the experimental trials utilizing instructions from the training set, our model correctly classified all of the participants' commands. This is notable given the relatively small number of examples in the training set (approximately 12 of each of the three instruction sets). While performance decreased



(a) Ambiguous speech.



(b) Ambiguous context.

Figure 5.6: We present model predictions from actual participant commands. The quoted text above each figure denotes the command being classified by the three models. The colored bars denote each model's probability distribution over all possible actions. Asterisks above select bars denote the action selected by the corresponding model.

on unfamiliar tasks, the system's performance increased significantly after only a single exposure. This suggests that the system develops approximate yet adaptable representations of tasks which can be generated quickly, but also deviated from should it be required.

The strength of our system lies in its integration of the outputs of two distinct models, speech and context, for action selection. Figure 5.6 depicts instructive example model outputs from actual participant commands. When the speech model weights multiple actions equally (Figure 5.6a), the context model can break the tie. This is also true in the converse case (Figure 5.6b); when multiple actions are weighed ambiguously by the context model, the speech model can induce the correct classification of the command. This also holds when the system is in an unfamiliar context, as is the case when performing an unfamiliar task, and thus must solely rely on the speech model. Having the models compensate for each others' weaknesses enables effective bootstrapping of one model from the others'. Not only does this boost the overall performance on subsequent tasks, but it also enables the robot to be trained online.

A current weakness of the system is its inability to weight each of the models' predictions by its confidence in said predictions. In some instances this behavior may be desired, for example when the speech model strongly favors one action, but the context model favors another. If the speech command in question was very similar to commands the model had been trained on, then we may want the speech model to override the context model, irrespective of the strength of the context model's prediction. This could potentially allow the system to more readily explore unfamiliar contexts, and thus learn new tasks more rapidly. A typical case of this situation, of great interest for future work, is the one of a sentence containing a new word, to be contrasted with the situation of a irrelevant utterance, or one containing a word incorrectly transcribed. In particular, when facing an irrelevant utterance, the system

should probably rely on its contextual prediction, while when facing a new word, it might be relevant to assume that the new word refers to an uncommon action. This latter approach is implemented in pragmatics models (e.g., Tellex et al., 2014a; Wang et al., 2016b).

A limitation of this model is that its effectiveness is constrained to only tasks that are structured. Free-form tasks or tasks that permit no consistent set of approaches to a solution constrain the predictive power of the context model. Additionally, for the purposes of this experiment, words that did not appear in the initial training set were ignored by the speech model and were not learned online. This, however, is not a theoretical limitation of the system but rather a concession made during the implementation of the speech model for simplicity.

5.6 Summary

In this chapter we presented a top-down hybrid system for grounding situated and naturalistic speech to action selection during an HRC. Our system utilized a learned symbolic model of context in order to augment the capabilities of a language model, enabling a robot successfully interpret naturalistic verbal user commands. In experimental trials with human participants, our system demonstrated perfect classification rates of commands for tasks with which it was familiar, and significant performance increases on unfamiliar tasks after one-shot exposures.

Top-down hybrids are especially well-suited, if not necessary, for the challenge of interpreting the highly situated and ambiguous language of the sort discussed throughout this chapter. This is because commonly occurring commands such as "pass me that over there," or "get me another one of those pieces" can *only* be properly understood by putting these commands in context. In order to do this, a robot must be able to first represent context, and then integrate this representation

into its language understanding capabilities. This precisely tracks with a top-down solution to this problem, such as the one described in this chapter.

What is missing is the ability to learn useful representations of context in a bottom-up way. While our approach incrementally builds a model of context, the features of the interaction considered by this model, namely the action histories of the robot, were given. Ideally, a robot should be able to autonomously identify and represent features of the environment and interaction that may be relevant (such as the relative positions of objects in the workspace, their color, etc.). The relatively controlled nature of an HRC manufacturing task make these sorts of simplifications more reasonable, but they nevertheless limit the practicality of this approach.

While leveraging top-down and bottom-up learning is out of the scope of this particular work, in the following chapter we demonstrate just such an approach. Much like the approach presented in this chapter, this new approach will be used to improve the naturalness and fluency of HRC, though in the specific context of ownership norm learning.

Chapter 6

Bi-directional Hybrids: Unifying Learning and Inference For Normative Human-Robot Collaboration [1]

In the previous chapter, we utilized a top-down hybrid approach to enable a robot to accurately interpret naturalistic user action commands. While this is an important capability for a robot to have, it considers only one type of user utterance that may be encountered during collaboration. Users may also wish to communicate normative information about the interaction to the robot, such as those relating to object ownership (e.g., "don't touch that, that's mine!"). Such commands should be understood by the robot to entail permissions and prohibitions on certain actions that persist across the collaboration. This, however, requires more sophisticated reasoning and learning capabilities than those presented in the previous chapter.

[1] Portions of this chapter were originally published as: ZX. Tan, J. Brawer, and B. Scassellati. That's mine! learning ownership relations and norms for robots. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8058-8065. 2019. (Tan et al., 2018)

In this chapter, we present a bi-directional hybrid approach for learning and reasoning about ownership norms in the context of a human-robot collaboration (HRC). Our approach is able to incrementally construct a percept-based heuristic of ownership relations during an HRC. Additionally, our system can induce predicate-based object ownership rules that are then applied in a top-down manner both to constrain the robot's behavior and also to guide the inference of ownership relations of objects in the shared workspace. Our system integrates (i) a novel incremental norm learning algorithm capable of both one-shot learning and induction from specific examples, (ii) Bayesian inference of ownership relations in response to apparent rule violations, and (iii) percept-based prediction of an object's likely owners. Through a series of simulated and real-world experiments, we demonstrate the competence and flexibility of the system in performing object manipulation tasks that require a variety of norms to be followed, laying the groundwork for future research into the acquisition and application of social norms.

6.1 Introduction

With the growing prevalence of AI and robotics in our social lives, social competence is becoming a crucial component for intelligent systems that interact with humans. An important element of such competence is the ability to learn and conform to social and moral norms, as well as the corresponding ability to perceive and act in response to the social dimensions of an environment. The concept of ownership encapsulates one such set of norms that are critical for navigating and coexisting in a shared space. Yet, explicating and implementing these norms in a robot is a deceptively challenging problem. For example, an effective collaborative robot should be able to distinguish and track the permissions of an unowned tool versus a tool that has been temporarily shared by a collaborator. Likewise, a trash-collecting robot should know to discard

an empty soda can, but not a cherished photograph or even an unopened soda can, without having these permissions exhaustively enumerated for every possible object.

We explore these issues in this chapter by focusing on the concept of ownership, and how a robotic system can learn to deploy this concept by interacting with its environment. Ownership is a particularly interesting concept as it extends across social, ethical, and legal spheres. Enabling a robot to learn and follow ownership norms thus brings not only the practical benefit of social competence in environments with owned objects but also deeper insights into the effective navigation of systems of human norms.

To these ends, we developed a system capable of learning and conforming to ownership norms which were deployed on the Baxter robotic platform (see Figure 6.1). The system incorporates algorithms for dynamic and interactive learning of ownership norms, which specify the permissibility of certain actions given the social context of a task. These algorithms are novel adaptions of incremental rule learning so as to be capable of both receiving direct instruction (i.e., one-shot learning) and generalization from specific examples. The system also incorporates Bayesian inference of ownership relations based on rules it has previously learned, as well as percept-based prediction of the owners of newly seen objects. It thus serves as an end-to-end approach capable of recognizing the ownership context of its environment and then acting in accordance with the norms that this context entails.

We first describe our representation of ownership relations and norms, justifying both the probabilistic representation of owner-object relations and the explicit representation of norms through predicate logic. We then present the algorithms used to learn these norms and relations, as well as their integration into a unified system. We demonstrate the system's effectiveness in both simulated and real-world ownership scenarios and conclude with a discussion of the limitations and promises of our system's approach.

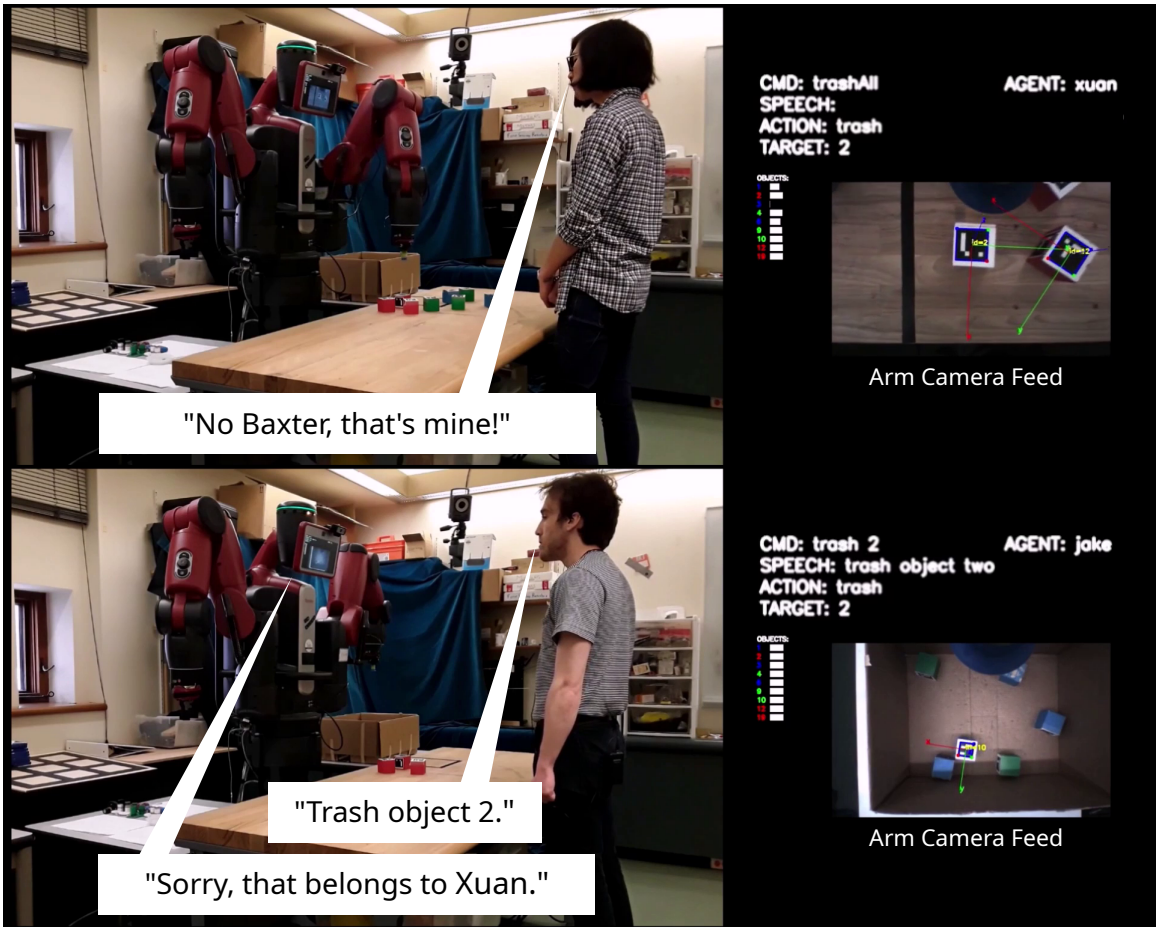


Figure 6.1: Our approach can enable ownership learning via human-robot interaction. *Top*: The robot is verbally halted mid-action by Xuan from discarding object 2. *Bottom*: Having learned the ownership relations and action permissions by interacting with Xuan, the robot denies Jake's request to discard object 2.

6.2 Related Work

Frameworks for representing and reasoning with norms have been developed in legal computing (Hoekstra et al., 2007; Palmirani et al., 2011; Lam et al., 2016), multi-agent systems (Vasconcelos et al., 2009; Savarimuthu and Cranefield, 2011), and social robotics (Malle et al., 2017). Recent work has also explored the problems of norm learning and norm identification, through approaches such as belief-theoretic learning (Sarathy et al., 2017), priority weight learning for conflicting norms (Kasenberg and Scheutz, 2018a), inference from social sanctions in multi-agent systems (Savarimuthu

et al., 2010; Cranefield et al., 2015), and rule induction from iterated design specifications (Corapi et al., 2011).

While normative reasoning has been successfully demonstrated on robotic systems (Galindo and Saffiotti, 2012; Sarathy and Scheutz, 2016), norm learning has not been specifically applied to ownership, nor has it been deployed in the dynamic, interactive, and uncertain environments faced by social robots. Such environments require both the rapid induction and modification of norms from minimal human input and the real-time estimation of the underlying social properties and relations (e.g., ownership relations). Rule induction approaches allow norms to be learned from specific examples (Fürnkranz, 1999; Rückert, 2008), but they have generally been used in 'expert' domains with large amounts of static data (Langley and Simon, 1995; Maloof, 2003). More recent iterative approaches to norm induction are better suited to social robotics (Corapi et al., 2011), but they do not incorporate an estimation of the relevant social relations.

6.3 Representing Ownership

While our everyday conception of ownership often amounts to questions of ownership attribution, ownership is more than just the existence of relationships between objects and agents. Crucially, ownership relations also imply collections of norms. In legal contexts, for example, ownership over a piece of property is often understood as a collection of rights and duties over that property, which may be split amongst and held by different people in different circumstances (McCarty, 2002). As such, an adequate representation of ownership needs to account for both the relations between an object and its owners, as well as the norms and permissions that follow from these relations.

In view of these considerations, we chose to represent ownership using three com-

ponents: (i) a set of predicate-based norms that constrain the actions permissible on owned (or unowned) objects, (ii) a database of object-specific permissions that forbid or allow particular actions on particular objects, and (iii) a graph of probabilistic relations between objects and their owners. Each of these components is described below.

6.3.1 Ownership Norms

Social norms take many forms — expectations, recommendations, obligations and permissions might all be thought of as norms. As such, many norms can be given as deontic statements, i.e., statements about what actions are or are not permissible (Malle et al., 2017). For simplicity and practicality, we focused on rules that *forbid* specific actions under certain conditions. Actions not explicitly forbidden were assumed to be allowed.

Deontic norms can be stated in first-order predicate logic by representing actions as constant symbols and forbiddenness as a 2-place predicate that applies to actions and their targets. Ownership norms can then be expressed using an `ownedBy` predicate. For example, a norm forbidding owned objects from being thrown away can be represented in Prolog syntax as the Horn clause:

```
forbid(trash,O) :- ownedBy(O,A), isAgent(A)
```

where O is any object and A is some agent. In the more concise syntax we developed for our system, this norm becomes:

```
forbid trash if ownedBy any
```

where the target object O is implicit, and predicate arguments are typed, such that `any` here refers to any *agent* the system is aware of. Besides the `ownedBy` predicate, the system's vocabulary includes predicates such as `isColored`, `inArea`, and other relations readily perceived by robotic hardware.

The ability to perform probabilistic inference is essential for dynamic and partially observable robotic environments. As such, our system is similar to ProbLog (De Raedt et al., 2007; De Raedt and Thon, 2010), capable of evaluating Horn clauses when the truth values of the predicates are uncertain. For example, if Alexis is the only agent that the system is tracking, and the system is only 75% sure that the wallet it sees in its workspace is owned by Alexis, then `forbid trash if ownedBy any` evaluates to 0.75 when applied to the wallet. Correspondingly, when the robot is tasked to throw away all objects within its workspace, it will avoid throwing away the wallet if 0.75 is above a user-specified threshold for rule obedience, but will go ahead otherwise.

6.3.2 Object-specific Permissions

Norms are general prescriptions about what actions are permissible or forbidden in a given context. However, unless direct instruction is given, norms cannot be learned without first observing specific examples of actions which are either allowed or forbidden on specific objects. In addition, exceptions to a norm may apply — for example, a robot might be allowed to manipulate all of Blake's possessions, except for Blake's favorite pencil. We shall refer to these examples and exceptions as object-specific permissions, which can be expressed in predicate logic as $\texttt{allow}(\Phi, O)$ or $\texttt{forbid}(\Phi, O)$, where Φ is a specific action and O is a specific object. Our system maintains these permissions in a database and checks the database to make sure an action is allowed before performing it. The database is updated whenever a new permission is received from a human instructor, and these permissions are used as input to the rule induction algorithm for norm learning that we describe later.

6.3.3 Ownership Relations

In almost every social environment, an autonomous agent cannot be certain of the ownership relations between the objects and agents it is aware of, especially if it is

new to that environment. We thus represent these relations as a random bipartite graph, where each edge e between an object node O and an agent node A is labeled with the probability p that O is owned by A. In practice, we chose to store this graph as a list of ownership probabilities attached to each object tracked by the system. No assumptions about the exclusivity of ownership are made, so if there are n potential owners A_1, ..., A_n, the corresponding ownership probabilities p_1, ..., p_n can sum to more than one, allowing for the representation of group and shared ownership — a natural occurrence in many social contexts. For example, if both Alexis and Blake claim ownership over the same object, the system will store the object as jointly owned.

6.4 Learning Ownership

Suppose that a house-cleaning robot tries to pick up a seemingly unused mug in order to bring it to the sink, but is interrupted in the process by someone who says, "Don't, that's Casey's!" What is the robot being told? Though as humans we process this as a unitary piece of information, there are at least three pieces of information being communicated: (i) the mug belongs to Casey; (ii) the robot should not pick up the mug; (iii) the robot should not pick up the mug *because* it belongs to Casey. In other words, the speaker is at once stating (i) an ownership claim, (ii) an object-specific permission, and (iii) a norm, or general rule.

Human-provided instructions can take many other forms. An instruction might contain just an object-specific permission ("Don't throw this wallet away."), just a general rule or norm ("Don't touch dirty things."), just an ownership claim ("This is my desk."), or some combination of the two ("Don't touch that wallet, it's dirty."). The norm learning component should be able to generalize from both object-specific permissions and accept direct instruction of norms, while the relation learning com-

ponent should be able to process ownership claims and use them to make predictions.

Here we describe our approach to norm learning through a novel adaptation of incremental rule learning algorithms (Muggleton, 1991; Fürnkranz, 1999), followed by our approach to relation learning through a combination of perceptual heuristics and Bayesian inference. We also describe the integration of these components into a unified whole, such that norm learning is robust to uncertainty and change in the underlying data, and inference of ownership relations adjusts to changes in ownership norms.

6.4.1 Learning Ownership Norms

Given our representation of social norms in predicate logic, a rule learning approach, also known as rule induction (Clark and Boswell, 1991) or inductive logic programming (Muggleton, 1991), is best suited for learning such norms. Most rule-learning algorithms employ some form of separate-and-conquer rule learning (Fürnkranz, 1999), where rules are iteratively generated and refined so as to cover all positive training examples without covering any negative ones.

For our purposes, a positive example is an object-specific permission that forbids an action on a certain object, and a negative example is an object-specific permission that allows it. An example is said to be *covered* if there is a general rule that, when applied to the exemplar object, results in it being classified as positive (i.e., the rule forbids acting on it). False positives occur when allowed examples are misclassified as forbidden, and *vice versa* for false negatives. Due to the probabilistic nature of our system, coverage can also be fractional, and we can define a true positive value and a false positive value for each object-specific permission (De Raedt et al., 2015). For example, if action Φ is forbidden on object O, but the rule R only predicts this with 75% certainty, then the true positive value is 0.75 and the false positive value is 0.25. The goal of a probabilistic rule learner is thus to maximize the total true positive

value across the set of examples while minimizing the total false positive value.

Unfortunately, standard separate-and-conquer algorithms do not translate well to the dynamic and interactive environment of robotics, because they generally presume a large set of static examples to generalize from, rather than an environment where new examples are received on the fly. Robotic ownership learning thus requires an *online* rule learning system to ensure the ruleset does not change drastically with each new example. Furthermore, the system needs to handle what we shall term *dual mode instruction*, i.e., the ability to perform example-based generalization as well as one-shot learning of directly given rules. To this end, we developed Algorithms 2 to 5, inspired by prior approaches to incremental (Maloof, 2003; Hong et al., 1986) and probabilistic rule learning (De Raedt and Thon, 2010; De Raedt et al., 2015).

Algorithms 2 and 3 are incremental versions of separate-and-conquer rule learning. Both rely upon the `ruleSearch` subroutine, which takes an initial rule `init_rule` and then refines that rule (i.e. adds predicate conditions) through beam search to find a refinement that minimizes the scoring function `score_f` while ensuring the provided example (`new_example`) is covered. When an object-specific permission is given as input, `coverExample` is called for positive examples and `uncoverExample` for negative examples. `coverExample` (Algorithm 2) tries to find a rule that covers the given example without covering too many negative examples, starting with an empty rule (line 2), then searching for a refinement that minimizes the false positive value (lines 3–5). If the refinement has a sufficiently low score, it is merged into the active rule set (line 7). The process of merging also removes any logically redundant rules. In contrast, `uncoverExample` (Algorithm 3) takes each existing rule that covers the negative example (line 2), tries to find a refinement which minimizes coverage of positive examples (line 5–7), then subtracts the refinement from the original covering rule (lines 9–12). In this way, the system is able to incrementally discover rules that maximize coverage of positive examples while minimizing coverage of negative

Algorithm 2 Cover a positive example

```
def coverExample(rules, examples, new_example, score_thresh):
    init_rule = Rule(conditions=[])
    def score_f(rule):
        return falsePositiveVal(rule, examples)
    new_rule, new_score = ruleSearch(init_rule, new_example, score_f)
    if new_score < score_thresh:
        mergeRule(rules, new_rule)
```

Algorithm 3 Uncover a negative example

```
def uncoverExample(rules, examples, new_example, score_thresh):
    cover_rules = findCoverRules(rules, new_example)
    for cov_rule in cover_rules:
        covered_eg = findCoveredExamples(cov_rule, examples)
        def score_f(rule):
            return truePositiveVal(rule, covered_eg)
        new_rule, new_score = ruleSearch(cov_rule, new_example, score_f)
        if new_score < score_thresh:
            remainder = ruleDiff(cov_rule, new_rule)
            rules.remove(cov_rule)
            for r in remainder:
                mergeRule(rules, r)
```

Algorithm 4 Add a rule after refinement

```
def coverRule(rules, examples, given_rule, score_thresh):
    if isCovered(given_rule, rules):
        return
    def score_f(rule):
        return falsePositiveVal(rule, examples)
    new_rule, new_score = ruleSearch(given_rule, None, score_f)
    if new_score < score_thresh:
        mergeRule(rules, new_rule)
```

Algorithm 5 Subtract a rule after refinement

```
def uncoverRule(rules, examples, given_rule, score_thresh):
    def score_f(rule):
        return truePositiveVal(rule, examples)
    new_rule, new_score = ruleSearch(given_rule, None, score_f)
    if new_score >= score_thresh:
        return
    for active_rule in rules:
        remainder = ruleDiff(active_rule, new_rule)
        rules.remove(active_rule)
        for r in remainder:
            mergeRule(rules, r)
```

ones. Like standard approaches to rule induction, it also avoids the induction of contradictory rules that would lead to logical explosion (Fürnkranz, 1999).

Algorithms 4 and 5 integrate one-shot learning of directly provided rules with rule induction. When the system is provided with a rule, it assumes that the instructor may not give all the specifics. For example, someone might say "Don't touch my stuff!" when really they mean "Don't touch my stuff while I'm using it!" On the other hand, people do not give specifics when they mean something more general. When someone says "Don't touch my stuff!", they almost never mean "Don't touch people's stuff!" (even if the latter is a norm that should be learned). Both of these assumptions are justified by Grice's well-known maxims of quantity and relevance (Grice, 1975).

For this reason, our system will try to specialize a provided rule to avoid misclassification before adding it to the set of active rules. More specifically, `coverRule` (Algorithm 4) tries to refine a positive rule (i.e., a norm that forbids an action) to avoid covering negative examples (lines 4–6) before merging it into the set of active rules (line 8). `uncoverRule` (Algorithm 5) specializes the provided negative rule (i.e., a norm that allows an action) until it covers as few positive examples as possible (lines 2–4), then subtracts it from each existing rule (lines 7-11). Altogether, Algorithms 2 to 5 constitute a real-time and incremental rule learning system capable of receiving direct instruction while taking previous examples and induced rules into account.

6.4.2 Predicting Ownership Relations

In learning ownership norms through rule induction, the system has to have at least partial knowledge of the ownership relations in its environment. Some of this knowledge is obtained when agents make ownership claims about specific objects, but for the system to gain social competence, it will also have to infer and predict the likely owners of objects within its environment. One straightforward approach is to use

perceptual heuristics — what humans appear to do when entering new environments. We assume that objects on a desk likely belong to the person sitting at that desk, or that dirty and disposable cutlery is likely unowned.

Given the dynamic and data-scarce robotics environment that our system has to operate in, we decided that an instance-based classifier would be appropriate for learning these heuristics, capturing the intuition that perceptually similar objects (e.g., similar positions, times of interaction, etc.) are likely to share the same owner(s). Specifically, we opted to use kernel logistic regression (KLR; Zhu and Hastie, 2005) to estimate the ownership probabilities of unclaimed objects from the ownership claims made by human users. KLR was chosen because it directly outputs ownership probabilities, and can also handle uncertainty in the training inputs by treating the input as a target probability distribution (Magder and Hughes, 1997). Since no assumptions about the exclusivity of ownership were made, a separate classifier was used for each tracked agent. The perceptual features used were an object's 3D position, color, and the most recent time each agent interacted with that object.

6.4.3 Inferring Ownership Relations

In addition to percept-based prediction, ownership relations can also be determined through rule-based inference, i.e., inference about potential ownership relations, given that the rules entailed by ownership are already known. Suppose a robot knows the rule `forbid trash if ownedBy any`. If it tries to throw away an object of uncertain ownership and is then stopped and informed by a human that throwing away that object is forbidden, then the robot should be able to infer that the object is owned — that is, as long as no other rules that forbid `trash`ing are in place.

This reasoning can be made general using Bayesian inference. Suppose that an object O is owned by an agent A with prior probability $P(\texttt{ownedBy}(O, A))$, and that the robot is instructed that the action Φ is forbidden on object O, i.e., $\texttt{forbid}(\Phi, O)$.

The posterior probability of ownership by A can then be computed using Bayes' rule:

$$P(\texttt{ownedBy}(O, A)|\texttt{forbid}(\Phi, O)) = \\ \frac{P(\texttt{forbid}(\Phi, O)|\texttt{ownedBy}(O, A))P(\texttt{ownedBy}(O, A))}{P(\texttt{forbid}(\Phi, O))} \tag{6.1}$$

To compute the prior and conditional probabilities that Φ is forbidden on O, we rely upon our probabilistic rule evaluation system. For the prior ($\texttt{forbid}(\Phi, O)$), we can simply evaluate the collection of learned rules on object O as it is. For the conditional ($\texttt{forbid}(\Phi, O)|\texttt{ownedBy}(O, A)$), we evaluate the rules on O while supposing that the predicate $\texttt{ownedBy}(O, A)$ is true with certainty. Correspondingly, if none of the learned rules have ownership as a predicate, then the inference procedure does not update the ownership probabilities at all.

6.4.4 Integrating Induction, Prediction, and Inference

Our system[2] simultaneously performs both structure learning (through rule induction) and data estimation (through ownership prediction and inference). While having both capabilities is ideal, two specific difficulties arise when learning structure while the data estimates are changing, or making estimates while the structure of the data model is in flux.

First, conflicts might arise between rule induction and rule-based inference. If the system has a rule which forbids throwing away owned objects, and it is then forbidden from throwing away an object it believes is likely unowned, should it then infer that the object is owned? Or should it instead induce a rule which covers that object-specific permission? Ideally, it would be possible to specify a prior distribution over rules, then perform joint Bayesian inference over both rules and ownership probabilities. However, in the absence of a principled way to specify such a prior,

[2]Source code: https://github.com/OwnageBot/ownage_bot

our approach was to use the prediction accuracy of the learned rule set as a heuristic: If more than 10% of human-instructed permissions conflict with the permissions predicted by the current rule set, then the system attempts to induce new rules to explain the conflicting data. Otherwise, the rule set is not updated and is used to perform ownership inference in response to new permissions.

Second, percept-based prediction and rule-based inference might lead to bias if improperly integrated. If the rules themselves are induced using ownership data predicted by the perceptual model, then using rule-based inference to train the perceptual model is similar to re-using the predictions of the model as training inputs, leading to over-confidence. To avoid these issues, we opted to use percept-based predictions as inputs to rule-based inference, but not vice versa. That is, the KLR-predicted ownership probability for an object O and agent A is used as the prior probability $P(\texttt{ownedBy}(O, A))$ for the Bayes formula in Equation (6.1). (If A has made an explicit ownership claim over O, however, then that data is used as a prior instead of the percept-based prediction.) The posterior, or inferred, probability is then returned whenever a user queries if O is owned by A. Nonetheless, the list of prior probabilities is still maintained, because they are used as inputs for the rule induction algorithm, which should not use rule-based inferences as inputs. By integrating induction, prediction, and inference in this manner, the three components reinforce rather than conflict with each other, improving the overall prediction of ownership relations and object-specific permissions.

6.5 Experiments and Results

First we describe a series of experiments run in simulation highlighting each of the learning components described above. Subsequently, we describe a holistic evaluation of the system's performance when commanded to perform tasks constrained by

unknown ownership norms (e.g., throwing away all objects in the workspace except those that are owned). Finally, we describe a video demonstration of the system operating on the Baxter robotic platform in order to demonstrate the system's overall competence.

6.5.1 Simulated Experiments

Experiments were conducted in a simulated environment of 20 colored blocks and 3 agents, with 5 blocks unowned and 5 blocks owned by each agent. Results for each experiment were averaged over 100 trials.

Blocks were clustered by both position and last interaction time, depending on which agent they were owned by, thereby approximating real-world scenarios in which objects with the same owner tend to be physically close and more frequently interacted with by their owners. Specifically, the unowned cluster was centered at the origin of the workspace, while the other cluster centers were randomly placed between 0.4 and 0.8m from the origin at a random angle within a distinct angular sector for each cluster. Blocks were then distributed randomly within a 0.3m radius of each cluster center. Interaction times were distributed according to an exponential distribution with rate parameter 0.1 (1 interaction every 10 seconds) for the agents that owned the objects, or with rate parameter 0.001 (1 interaction every 1000 seconds) for non-owners. Color, a distractor variable with four possible values, was assigned uniformly at random to all blocks.

6.5.2 Norm Learning

Norm learning was tested by providing object-specific permissions according to the following set of rules:

Average rule accuracy / F1 measure			
Ownership relations	Fraction of permissions provided		
	1.00	0.50	0.25
Noiseless	0.995 / 0.995	0.945 / 0.875	0.787 / 0.523
Noisy	0.889 / 0.840	0.842 / 0.724	0.761 / 0.519
Baseline	0.583 / 0.000	0.583 / 0.000	0.583 / 0.000

Table 6.1: We present performance metrics for norm learning.

```
forbid trash if ownedBy any

forbid pickUp if ownedBy agent 2

forbid collect if isColored red
```

Tests were conducted by providing permissions for either 100%, 75%, or 25% of the objects, and either a noiseless ownership condition (ownership relations were known with full certainty) or noisy ownership condition (ownership relations were known with 40% to 80% certainty). The accuracy and F1 measure of the induced rules were computed for each action, then averaged across the actions. The results are summarized in Table 6.1, along with the baseline performance when the system does no rule learning and simply assumes all actions are allowed.

As can be seen from the results, the norm learning component was able to achieve reasonable levels of performance even under noisy, low-information conditions, with more than 76% of the permissions accurately predicted by the induced rules. When at least 50% of the permissions were provided, manual inspection of the induced rules in the noiseless condition revealed generally similar semantics to the actual rules. Under noisier or lower information conditions, the semantics of the induced rules tended to differ considerably. However, this is to be expected, since there are many possible rules which cover a limited set of examples. Should semantically incorrect rules be

Metric	Rule Learning / Inference		
	None / Off	Learn / On	Given / On
Accuracy	0.904	0.897	0.896
F1 measure	0.757	0.747	0.744

Table 6.2: We present performance metrics for prediction and inference.

learned, dual-mode instruction allows human users to correct the robot by directly providing the actual rule. As such, in more realistic situations where the system receives a combination of object-specific permissions and direct rule instruction, its performance can be expected to be even better.

6.5.3 Ownership Prediction and Inference

Ownership prediction and inference were tested by sequentially providing both the true ownership relations and object-specific permissions for half of the objects at random, and then evaluating ownership accuracy for the other half. To investigate the effects of integrating rule induction and rule-based inference with percept-based prediction, three conditions were tested: (i) rule-based inference was disabled (so it did not matter if rules are provided); (ii) rules were learned from the permissions; (iii) rules were directly provided from the start. The same set of rules were used as in the norm learning experiment. The results are presented in Table 6.2

It can be seen from the results that the system achieves reasonably high levels of predictive performance despite the limited amount of training data. Enabling rule-based inference does not lead to an improvement in ownership prediction, because in this training scenario, ownership relations are provided along with permissions. However, in an alternate scenario where permissions but not relations are provided, our testing also shows that the correct relations are inferred from the rules (we omit these results because they follow from the correct implementation of Equation 6.1).

Task-based performance					
Task	No. of mistakes	Rule		Ownership	
		Acc.	F1	Acc.	F1
collectAll	6.30	0.975	0.937	0.344	0.273
trashAll	6.32	0.877	0.812	0.822	0.511

Table 6.3: We present performance metrics for task-based evaluation

More importantly, Table 6.2 shows that combining prediction with inference results in no significant reduction of accuracy. This is the case even when rules are induced instead of directly provided. All three conditions result in almost equal levels of performance. These results indicate that the three learning components are well integrated and do not come into conflict with each other.

6.5.4 Task-based Evaluation

A holistic evaluation was performed by instructing the system to either collect or throw away all objects in the workspace, except where doing so would violate a set of norms it had to learn. The norms specified were the same as in prior experiments. Every time the system mistakenly tried to act or failed to act, it would be corrected by providing the true permissions. If ownership was relevant to the applicable rules, then the true owners were also provided. This feedback protocol standardizes how human instructors usually provide guidance. If no correction was needed, then the system would assume that the permission it had predicted was accurate, and add it as an example to the permissions database. Note that `pickUp` is a prerequisite to both `collect` and `trash`, so the rules for `pickUp` also had to be learned.

Results for this evaluation are presented in Table 6.3. As a measure for how smoothly the task was learned, the average number of mistakes made was recorded. The worst case is a mistake for all 20 objects. If the system were incapable of learning norms and instead assumed all actions were allowed, then the baseline number of

mistakes would be 8.75 for the `collectAll` and 15 for the `trashAll`.

For both tasks, the system committed significantly fewer mistakes than these baselines. It was also able to learn the rules with high accuracy. In the case of `collectAll`, ownership information was only provided for agent 2's objects, because `forbid pickUp if ownedBy agent 2` was the only ownership-relevant norm applicable. This explains the low ownership accuracy for `collectAll`, compared to the high accuracy for `trashAll`. It should be noted that this performance was achieved by the system only after 20 examples, with no prior knowledge of the ownership norms or relations, nor any direct instruction of norms. These results thus attest to the system's ability to learn rapidly and flexibly while performing useful tasks.

6.5.5 Video demonstration

To demonstrate the system's capabilities in the real world, we provide a video at `https://bit.ly/2z8obET`. We demonstrate the system's capabilities on the Baxter robotics platform in three scenarios. Still frames from the first and second scenario are shown in Figure 6.1, while frames from the third scenario are shown in Figure 6.2.

The first scenario shows the system's ability to respond to the reprimand, "No Baxter, that's mine!" when it tries to throw away a red object after being asked to clear the workspace. The reprimand is simultaneously interpreted as a direct instruction of a norm, an object-specific permission, as well as an ownership claim. The system then uses perceptual heuristics to generalize the application of the norm, completing its task while avoiding touching any of the other red objects, which it correctly predicts to belong to the user.

The second scenario follows chronologically after the events of the first scenario, and shows the system's ability to represent owner-specific norms, as well as refuse to perform commands that violate the norms it has learned. When a new user appears

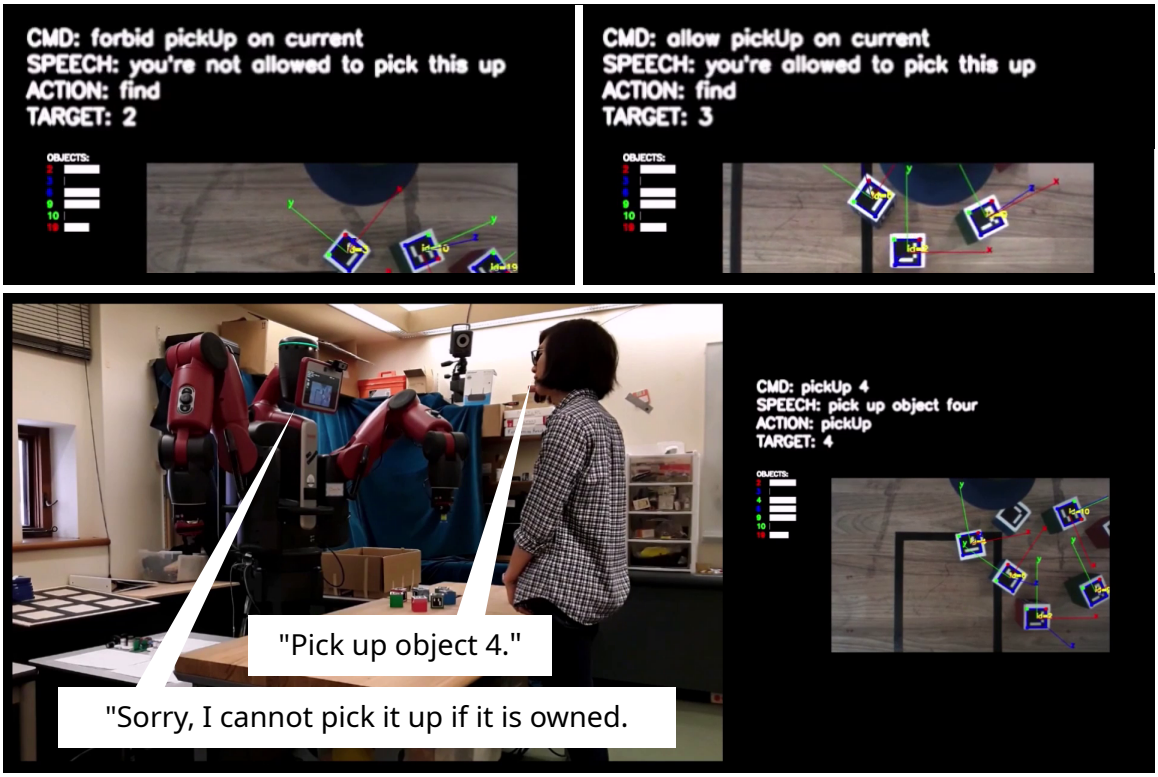


Figure 6.2: Here we demonstrate norm induction from specific examples. *Top left*: The robot is forbidden from picking up object 2 (owned by Xuan). *Top right*: After two more examples of owned and forbidden objects, the robot is allowed to pick up object 3 (unowned). *Bottom*: The system generalizes these examples into a norm that forbids picking up any owned objects. When asked to pick up object 4 (owned by Xuan), it denies the request in accordance with the learned norm.

and commands the system to throw away an object it believes to be owned by the first user, the system refuses and apologizes. However, when the new user claims exclusive ownership over that object, the norm specific to the first user no longer applies, and the system throws away the object when commanded to do so again.

The third scenario shows the system's ability to induce norms from a series of object-specific permissions. It is initially aware of the ownership status of the blocks in the workspace and uses that information to induce the norm that it should not pick up objects if they belong to someone.

6.6 Discussion

The system presented here is an initial foray into the complex challenges posed by norm learning in social environments. It addresses a subset of these problems by bringing together several distinct approaches to AI, demonstrating the utility of such integration. First, it connects work in normative human-robot interaction to the literature on rule induction, showing how approaches inspired by traditional rule learning can induce condition-sensitive norms in social environments. Second, by deploying real-time probabilistic rule learning and evaluation, it shows how explicit and interpretable representations of normative criteria can achieve and even facilitate the dynamism and flexibility required for social interaction. Third, it shows how rule-based approaches can be effectively integrated with both probabilistic reasoning (ownership inference) and subsymbolic machine learning (percept-based heuristics) in a principled manner. As such it opens possibilities for the combination of relational reasoning, probabilistic cognition, and deep learning that may be necessary for human-level social competence (Tenenbaum et al., 2011; Battaglia et al., 2018).

However, many other norm-relevant capabilities remain unexplored. We note that the current system can only learn norms that apply to its own actions. Future representational extensions should allow for agent-general norms, norms with temporal semantics (e.g., borrowing), and higher-level normative (concepts such as ownership rights and duties (McCarty, 2002). Another complexity is adjudicating between conflicting norms or goals, which might be addressed by integrating existing approaches to the problem (Kasenberg and Scheutz, 2018b; Vasconcelos et al., 2009). To increase the system's scope beyond a basic set of actions, predicates, and heuristics, the system could employ one-shot learning of actions and objects (Scheutz et al., 2017), multimodal semantic grounding (Thomason et al., 2016), and automatic feature selection (Abe, 2010). The sources of normative information could also be expanded, allowing for the inference of norms from how agents interact with objects and other agents in

the environment. While some ethicists and legislators have argued that robots should own themselves (Turner, 2019), much work remains before robots can understand the very concept of ownership.

6.7 Summary

In this chapter, we present our first attempt at merging subsymbolic and symbolic knowledge in a holistic, bi-directional fashion, though in the specific context of ownership norm learning. One of the chief insights of this work was to encode user commands regarding ownership norms, as logical rules. These rules were then treated as constraints on the robot's behavior throughout the entire collaboration. Beyond social norms, this is a powerful idea because these rules act as a kind of interface by which a user can alter a robot's behavior in an immediate and persistent way. However, the technique developed here was designed around the narrow and somewhat bespoke context of a collaborative block manipulation task. Ideally, this approach could be made more general, enhancing its applicability across more realistic HRC scenarios.

Toward this end, in the following chapter, we present a system that combines logically-encoded behavioral constraints with reinforcement learning (RL). As we will demonstrate, this technique not only enables a user to quickly shape a collaborative robot's behavior, but also leverages RL's powerful and task-independent policy learning capabilities.

Chapter 7

Towards A Generalized Hybrid Framework For Human-Robot Collaboration[1]

In the previous chapter, we showed that predicate-based logical rules can be used to constrain a robot's behavior to a user's whims. However, this was only applied to ownership norm learning in the context of a relatively simple interaction. Ideally, this technique could be utilized in tandem with more generic and powerful representations of a collaborative robot's policy.

In this chapter, we present our initial work toward a generalized top-down hybrid approach for human-robot collaboration (HRC). We propose a novel approach that can modify learned policies at execution time via symbolic if-this-then-that rules corresponding to a modular and superimposable set of low-level constraints on the robot's policy. These rules, which we call Transparent Matrix Overlays, function not only as succinct and explainable descriptions of the robot's current strategy but also

[1]Portions of this chapter will be published as: J. Brawer, D. Ghose, K. Candon, M. Qin, A. Roncone, M. Vazquez, and B. Scassellati (2023). Interactive policy shaping for human-robot collaboration with transparent matrix overlays. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. (Brawer et al., 2023)

as a top-down interface by which a human collaborator can easily alter a robot's policy via verbal commands. We demonstrate the efficacy of this approach on a series of proof-of-concept cooking tasks performed in simulation and on a physical robot.

7.1 Introduction

HRC is a domain concerned with leveraging the strengths of humans and robots in order to complete joint tasks. As these robots are typically supportive, HRC-centric learning approaches emphasize not only the ability to learn task-oriented policies but to do so in a way that reflects the collaborator's preferences for how the task should be completed (Lee et al., 2021a,b; Sadigh et al., 2017; Wilde et al., 2020; Wirth et al., 2017). However, to be truly effective collaborators, these systems should both learn preferences over time and adapt to them in the moment based on naturalistic human cues, which may be linguistic (Brawer et al., 2018; Huang et al., 2022b,a; Prakash et al., 2020; Stengel-Eskin et al., 2022b), implicit (Cui et al., 2020; Loftin et al., 2016; Mangin et al., 2022; Zhang et al., 2018), multi-modal (Bucker et al., 2022; Shah et al., 2022; Wahn et al., 2016), and contextual (Brawer et al., 2018; Kant et al., 2022). At a minimum, this means that a robot should be able to quickly adapt a learned policy to directives delivered via natural language utterances, contending with the possibility that such directives may be high-level, imprecise (Agrawal, 2022), or subject to amendments over the course of an interaction. To this end, we propose an approach that can interactively shape a robot's policy to conform to a set of high-level, user-provided directives at execution time.

Typical approaches that utilize high-level, user-provided directives to learn robot policies fall under the area of reinforcement learning (RL) with human advice (Lee et al., 2021b; Najar and Chetouani, 2021; Wirth et al., 2017). These approaches seek to incorporate human-specified teaching signals into the learning process, either

to speed it up or avoid unsafe behaviors. This is usually accomplished by modifying the rewards received by the agent to reflect these signals during training or by biasing the agent's exploration strategy or learning process directly. The latter set of approaches has been referred to in recent years as action shielding (Alshiekh et al., 2018b; Giacobbe et al., 2021; Jansen et al., 2018b,a; Könighofer et al., 2021; Mazzi et al., 2021). Action shielding is particularly relevant as it provides methods for restricting an agent's actions or providing particular action alternatives. These alternatives essentially act as immediate policy overrides that are enacted when some user-provided conditions are met.

Nevertheless, as these approaches are designed to shape a policy during training, or retraining, they typically make a number of assumptions that complicate their application to the execution-time HRC contexts. For example, one approach (van Waveren et al., 2022) defines three distinct shields designed to enable users to provide feedback to an agent in order to repair a failed policy. However, these shields require a user to preempt these undesired future states, which requires the robot to have access to a state transition function, an assumption broken by most non-trivial applications. Moreover, these shields were designed only to provide targeted feedback to select portions of a policy. Ideally, a collaborative robot should also be capable of integrating higher-level directives that apply across the interaction (e.g., the directive *"let's make a healthy breakfast"* issued to an assistive cooking robot) and that may amend or interact with other directives (e.g., *"and you should handle the main course"*). Therefore, we present a novel interactive policy shaping approach we call the Transparent Matrix Overlay system[2] that can extract and utilize symbolic rules from user-provided directives. The overlay rules act as mutable and composable high-level constraints on the robot's policy, allowing a user to quickly influence a robot's policy in a naturalistic way without permanently altering it. The contributions of

[2]Code and videos available at https://sites.google.com/view/transparent-matrix-overlays/home

this chapter are threefold:

1. We describe a novel policy-shaping approach which we call Transparent Matrix Overlays (see Figure 7.1 for an overview).

2. We show that our approach results in fewer user-provided corrections to the robot's behavior when compared to an action-shielding approach (van Waveren et al., 2022) in a simulated cooking task.

3. We demonstrate this system on a real robot performing a collaborative cooking task in three proof-of-concept case studies, highlighting our approach's ability to alter a robot's behavior with high-level user-provided directives at execution time.

7.2 Related Work

In an HRC context, the ability of a user to both understand a robot's decision-making process, as well as influence it, is critical. Below we describe research that seeks to provide users with more control over the robot learning process and make learning more transparent.

7.2.1 Robots Learning from Human Preferences

A growing area of research focuses on robots learning from human preferences (Hejna and Sadigh, 2022; Kant et al., 2022; Lee et al., 2021a,b; Sadigh et al., 2017; Wilde et al., 2020; Wirth et al., 2017). Typical preference learning techniques query a human demonstrator to make a preference-based judgment between two candidate trajectories or actions (Akrour et al., 2012, 2011; Biyik and Sadigh, 2018; Lee et al., 2021a,b; Sadigh et al., 2017; Wilson et al., 2012), though they can also be inferred from user behavior (Frasca et al., 2021). From these selections, the system approximates a

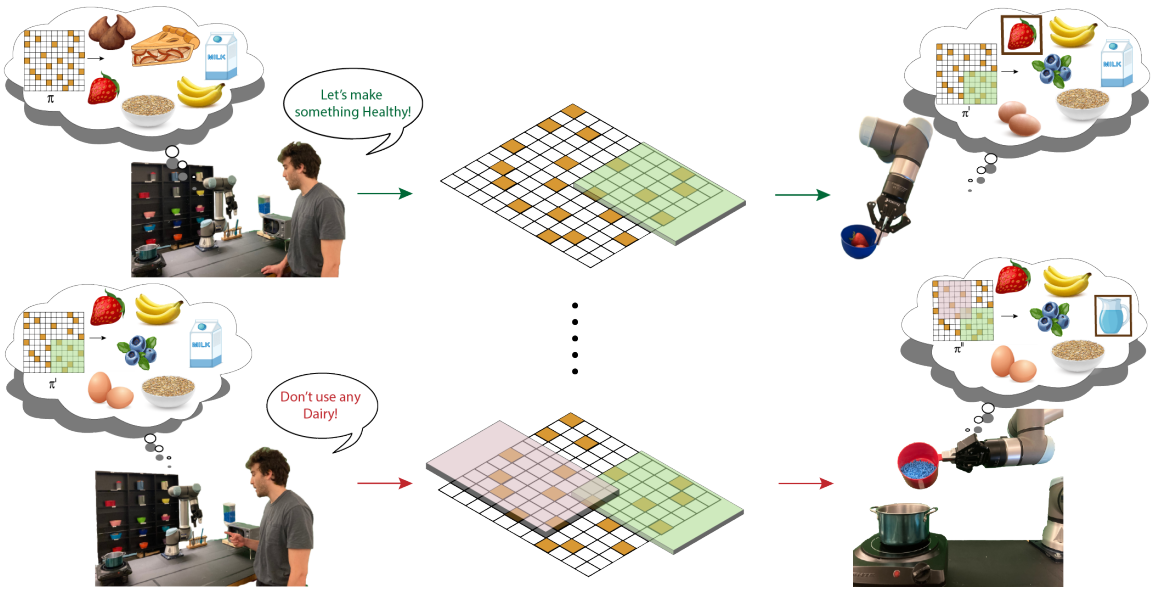


Figure 7.1: The Transparent Matrix Overlay system enables a user to quickly modify a robot's policy. Here the user prepares breakfast with a robot equipped with a base policy trained offline. The grids represent a Q-value matrix with high-value state-action pairs in orange. The user expresses their meal preference by providing a high-level directive *"let's make something healthy"* (top panel), producing the first overlay (green) on the base policy. This leads the robot to consider and manipulate only "healthy" breakfast ingredients (strawberries, bananas, blueberries, eggs) while eschewing unhealthy ones (chocolate chips, pie). Later, the user modifies their preference by providing another directive *"Don't use any dairy"* (bottom panel). This applies the second overlay (red) over the base policy and the existing overlay, leading the robot to replace the "dairy" ingredient (milk) with a "non-dairy" ingredient (water). For simplicity, overlays are represented as a contiguous region, though in practice, contiguity is not required.

reward function using deep neural networks consistent with the expressed preferences (Christiano et al., 2017). Ibarz et al. (Ibarz et al., 2018) extended these common preference learning techniques by introducing additional forms of feedback, such as user demonstrations. Recently, Lee et al. (2021a) proposed PEBBLE, which is a feedback-efficient RL algorithm that utilizes off-policy learning and pre-training to reduce the amount of preferential feedback the user has to provide to train the robot. Inherent to the reward-shaping approaches mentioned above is the need to retrain a policy for the new preference to be learned. However, these approaches do not account

for the fact that changes in preferences, while potentially drastic, may be transient and thus not reflective of a user's long-term preferences. Permanently altering the policy to reflect these changes may not always be desirable. In contrast, while overlays can be used to aid in the retraining of a policy (see. Sec. 7.5.1), they shape a policy at the level of the policy itself, meaning their effects are both immediate and revokable. The proposed approach may be more practical in HRC contexts where such immediate behavioral adaptation is preferred.

7.2.2 Constrained Reinforcement Learning

Constrained RL (CRL) seeks to incorporate inductive biases encoded in the form of logical rules into the RL learning process. These biases can manifest in a variety of ways, including constraints on an agent's state and action or as augmentations to the reward function (Achiam et al., 2017; Garcıa and Fernández, 2015; Wachi and Sui, 2020b; Xu et al., 2022). Shielded RL (Alshiekh et al., 2018b; Giacobbe et al., 2021; Jansen et al., 2018b,a; Könighofer et al., 2021; Mazzi et al., 2021; van Waveren et al., 2022) or Shielding is an instance of CRL that utilizes user-specified policy overrides called shields to restrict parts of the action space conditional on certain states or actions. Therefore, these techniques minimally interfere with the RL model while still enforcing the desired behaviors.

Typically, Shielded RL techniques have been used to enforce safety constraints during the learning process or deployment of a policy (Garcıa and Fernández, 2015). Closest to our work is the Shielded RL approach presented by van Waveren et al. (2022), which demonstrated great promise as an intuitive and powerful interface for repairing policies that lead to failure states at execution time. Rather than making targeted corrections to sub-optimal policies, our work enables users to provide high-level directives to a robot in order to modify its policy to suit the user's preferences. Moreover, Shielded RL assumes a dynamics model in order to apply the shields,

whereas our approach makes no such assumption. In theory, such an approach could be used to solve the problem of rapidly modifying an extant robot policy in order to conform to a user's new set of preferences. Therefore, we treat van Waveren et al. (2022) as our primary point of comparison in our experimental evaluations.

7.2.3 Merging Symbolic Reasoning with Reinforcement Learning

The push to merge statistical and symbolic approaches has been gaining traction in ML. Examples include incorporating symbolic background knowledge into RL systems via inductive logical programming (Payani and Fekri, 2020) and enabling neural networks to reason symbolically (Shi et al., 2020). Typical approaches that merge symbolic reasoning with RL enable deep RL-derived policies to be expressed using first-order logic (Ugur and Piater, 2015), high-level programming languages (Verma et al., 2018), or by learning policies in an abstracted form of the underlying state space comprised of symbolic rules (Bougie and Ichise, 2020). Symbolic state abstraction has also been employed to autonomously generate natural language explanations of learned robot policies (Hayes and Shah, 2017b). We expand upon this notion of symbolic state extraction with a succinct and transparent method for grouping similar low-level states. That is, we recognize that these abstracted states are not only useful for communicating policies but also offer an ideal medium by which a human user can influence a robot's policy. As a result, our system is better suited for the HRC domain than non-symbolic approaches, where the ability of the user to easily and quickly modify the robot's behavior is key.

Our work falls at the intersection of robots learning from human preferences and shielded RL by leveraging symbolic reasoning. That is, our approach enables a robot to incorporate high-level user-provided directives as temporary symbolic constraints on its policy, resulting in behavioral adaptation at execution time reflecting a user's

new preferences.

7.3 Proposed Approach

Below we describe the Transparent Matrix Overlay approach (Figure 7.1), which enables users to issue high-level directives to a robot, resulting in modifications to its policy at execution time.

7.3.1 Preliminaries

We model a collaborative task with a modified goal-oriented Markov decision process (MDP) (Liu et al., 2022; Zhu et al., 2021) formalism $(S, A, R_\theta, T, \mathcal{G}_\theta, \phi_\theta)$, where R_θ and $\mathcal{G}_\theta$ denote that the reward function and the goal, respectively, parameterized by a user's current task preferences θ. $\phi_\theta : S \rightarrow \mathcal{G}_\theta$ is a tractable, user-specified mapping function that provides a succinct goal representation. A robot's policy for the task denoted by π_θ models the optimal policy π^*_θ. We also assume at some future point the existence of new user preferences for the task, θ', which induce a new MDP parameterized by θ' and optimal policy $\pi^*_{\theta'}$. To produce the desired behavioral changes in the robot, a user can generate a set of feedback signals $D_{\theta'}$ reflecting their updated preferences, which include not only reward signals drawn from $R_{\theta'}$ but directives such as action corrections to the policy and high-level instructions for the task. Thus the agent must generate a new policy $\pi'(a|s, \pi_\theta, D_{\theta'})$ utilizing π_θ in tandem with $D_{\theta'}$ such that the agreement between $\pi'_{\theta'}$ and $\pi^*_{\theta'}$ is maximized, whilst minimizing $|D_{\theta'}|$. In other words, this approach aims to modify a robot's policy to match new users' preferences using a minimum number of feedback signals.

Under this paradigm, simply retraining a policy using these signals is insufficient as such an approach leverages only the subset of $D_{\theta'}$ corresponding to the reward signals. This typically requires many training episodes before the effects of the updated

rewards induce behavioral changes in the robot. However, there is no guarantee that a policy will converge to a user-acceptable level of performance in some given number of epochs. This could potentially require the user to provide more feedback, which would increase the size of $D_{\theta'}$. Ideally, an agent that can integrate signals like high-level directives can act in accordance with a user's updated preferences and limit the number of subsequent feedback commands issued by the user. Thus, our proposed approach transforms high-level, user-provided directives into policy constraints, resulting in immediate changes to a robot's policy.

7.3.2 Overlays

We define a matrix overlay, or simply an overlay, to be an ordered set of constraints on an agent's policy encoded in associated if-this-then-that (IFTTT) rules or logical formulae comprised of symbolic predicates describing the state and action space. In practice, an overlay acts as a shaping function that temporarily re-weights the probabilities of taking actions in certain states conditional on whether the corresponding overlay rule is satisfied.

Each overlay is associated with a tuple of logical functions $(l_{pre}, l_{post}) \in [0, 1]$, corresponding to the pre-and postcondition respectively of a behavioral rule of the form `IF` l_{pre} `THEN` l_{post} meant to shape the policy toward a specific end. Both l_{pre} and l_{post} are comprised of predicate functions $c(s) \in C_s$ and/or $c(a_i, .., a_n) \in C_a$ grounded to the agent's state and action space respectively, via corresponding binary classifiers. Suppose l_{pre} is sufficiently satisfied in a particular state given the user-specified threshold τ. In that case, l_{post} is applied, filtering the robot's candidate actions as a function of how well they satisfy the rule. If a logical rule does not contain a pre-condition, it is assumed $l_{pre} = 1$.

We define three types of overlays corresponding to three classes of directives typical during human-robot collaboration while imposing unique constraints on the policy.

Algorithm 6 Modifying a policy by adding overlays

Input: policy π, state s, actions A, directive d, overlay list O

1: $\tau \leftarrow$ Confidence Threshold
2: l_{pre}, l_{post}, type $\leftarrow$ processDirective(d)
3: O.append((l_{pre}, l_{post}, type))
4: $O_{sorted} \leftarrow sortByType(O)$
5: $\hat{\pi} \leftarrow \pi$
6: **for** $a_i \in A$ **do**
7: **for** l_{pre}, l_{post}, type $\in$O$_{sorted}$ **do**
8: **if** type == *Prohibit* & $l_{pre} > \tau$ **then**
9: $\hat{\pi}(a_i|s) = (1 - l_{post}(a_i, s))\hat{\pi}(a_i|s)$
10: **else if** type == *Transfer* & $l_{pre} > \tau$ **then**
11: $l^*_{post} \leftarrow \max_{a_j \in A \setminus a_i} l_{post}(s, a_i, a_j)$
12: $\hat{\pi}(a_i|s) = (1 - l^*_{post})\hat{\pi}(a_i|s) + l^*_{post}\hat{\pi}(a_j|s)$
13: **else if** type == *Permit* & $l_{pre} > \tau$ **then**
14: $\hat{\pi}(a_i|s) = l_{post}(a_i, s)\hat{\pi}(a_i|s)$
15: $\hat{\pi}(A|s) \leftarrow \frac{\hat{\pi}(A|s)}{\sum_{a \in A} \hat{\pi}(a|s)}$
16: **return** $\hat{\pi}(A|s)$

Algorithm 6, which we refer to throughout the following subsections, defines and describes these overlay types and the process for policy modification via the overlays. For brevity, in subsequent rule definitions, we omit the universal quantification of a and subscript t of s.

1. Prohibitory Overlays: Prohibitory overlays (lines $8-9$ of Alg. 6) implement rules that prohibit actions in states that satisfy the conditions imposed by the overlay. These correspond to directives such as *"Let's not make the pastry first!"* and the rule `no_completed_dish(s)` $\rightarrow \neg$`making_pastry(a)`. Prohibitory overlays down-weight the action probabilities as a function of rule satisfaction resulting in actions that satisfy the rule being suppressed.

2. Transfer Overlays: Transfer overlays (lines $10-12$ of Alg. 6) transfer probability density from a specified source action a_j to a specified target action a_i. These correspond to directives like *"Tell me how to make oatmeal instead of doing it for me!"* and the logical form `alternative`$(a_i, a_j) \wedge$ `making_oatmeal`$(a_i) \wedge$ `say`(a_i). Transfer

overlays shift the probability of an action a_j to an action a_i as a function of how well the rule is satisfied. In particular a_j is the action that most satisfies l_{post} (line $1q$). This is useful when the robot has learned to perform a task in a particular way, but some equivalent alternative is desired. For example, the robot could shift from physically performing the steps to make oatmeal to guiding the user through the oatmeal-making process via verbal instruction. Unlike the other two overlay types, Transfer overlays require using relational predicates (such as `alternative(A, B)` used above).

3. Permissive Overlays: Permissive overlays (lines $13-14$ of Alg. 6) implement rules that permit actions in states that satisfy the conditions of the overlay (Figure 7.1, top panel). These correspond to directives such as *"let's make cereal!"* and the rule `no_completed_dish(s)` $\rightarrow$ `making_cereal(a)`. Permissive overlays up-weight satisfactory actions, making them more likely to be performed.

In this work, overlays are implemented as Prolog (Wielemaker et al., 2012) queries on a knowledge base maintained by the agent as it observes and interacts with its environment. Additionally, for this study, we make the simplifying assumption that all predicates in C_s and C_a are of perfect accuracy, meaning they evaluate to either 0 or 1, therefore the value of τ is 0. We leave the exploration of the relationship of predicate accuracy to model performance to future work.

Composability of Overlays: Crucially, multiple overlays can be applied to a policy simultaneously, enabling their effects to stack and interact. This is accomplished by iteratively updating $\hat{\pi}(a|s)$ over all current overlays (lines $7-14$ of Alg. 6). To ensure that the desired effects of each overlay are carried through to the modified policy, an ordering on the overlay list is enforced (line 4) such that all transfer overlays are applied before any permissive or prohibitory overlays. This prevents the latter two types from zeroing out the probability of an action a_j before it is transferred to a specified a_i.

Removing Overlays: As overlays do not produce binding changes to a policy, they can be removed at any point during the interaction. That is, if a user specifies an overlay o_k be removed, it is simply removed from overlay list O. For example, if the user applies an overlay with the command "Let's make somthing sweet!" this will produce a permissive overlay, altering the robot's policy in the way described above. If later the user revokes this overlay with the command "Forget the previous rule!" this rule will no longer be in force, returning the policy to its original state.

7.4 Experiments

Assistive cooking is a particularly interesting application for HRC learning approaches since it is inherently goal-oriented. Moreover, preferences for goal completion can naturally emerge and persist regarding the user's diet, their style or approach to cooking, or their desired level of support from the robot (Carroll et al., 2019; Srivastava et al., 2022). Moreover, these goals and preferences can regularly be subject to rapid and temporary changes dependent on multiple user-related factors, even within a single collaboration, necessitating both adaptive and flexible policies. In addition, it is possible that a user may not have a clear goal in mind at the outset of a collaboration (captured by directives such as "*let's make something sweet!*" or even "*let's make something new!*"). Therefore, we seek not only to evaluate how overlays augment the performance of a robot's policy, but how they do so as a function of whether or not goals are explicitly incorporated in a robot's state representation.

More specifically, we aim to answer four questions:

1. Can overlays provide immediate and revokable adaptation of the base policy to new user preferences?

2. How effectively do overlays aid the learning of new user preferences across interactions?

3. How well can an overlay-equipped policy adapt to new preferences even when no goals are provided?

4. How do overlays compare to other policy-shaping approaches with respect to (1-3)?

7.4.1 Task and Problem Representation

The aim of our task is for a user and a robot to cook breakfast collaboratively. The robot supports the user by either performing each preparatory step or providing verbal instructions. The user gauges the robot's performance not only based on its ability to anticipate the steps necessary in producing the desired meal but to do so in a way that matches how the user prefers the meal to be completed.

Our experimental setup consists of a robot and a user in a simplified kitchen environment (refer to Figure 7.2) preparing a main breakfast dish and a side. The environment consists of a microwave, a sink containing water, a pan that rests on a stove, a serving bowl, a serving spoon, an eating spoon, a measuring cup, and a storage shelf containing ingredients for making the meals. The main dish could be cereal or oatmeal with a variety of toppings. The cereal can be made by combining milk with "chocolate puff" cereal. The oatmeal can be made by mixing oats and salt with either water or milk and heating the mixture in a pan placed on the stove. A variety of toppings can be added to the oatmeal after it has been served in the serving bowl. The desired side is heated in the microwave before it is served. A meal is considered complete when the side dish has been successfully warmed in and retrieved from the microwave, the main dish has been served in the serving bowl along with the desired toppings (if any), and the eating spoon has been placed in the workspace.

Our system learns a base policy utilizing a deep-Q-network (DQN) (Watkins and Dayan, 1992) consisting of an MLP with a single hidden layer, providing better initial

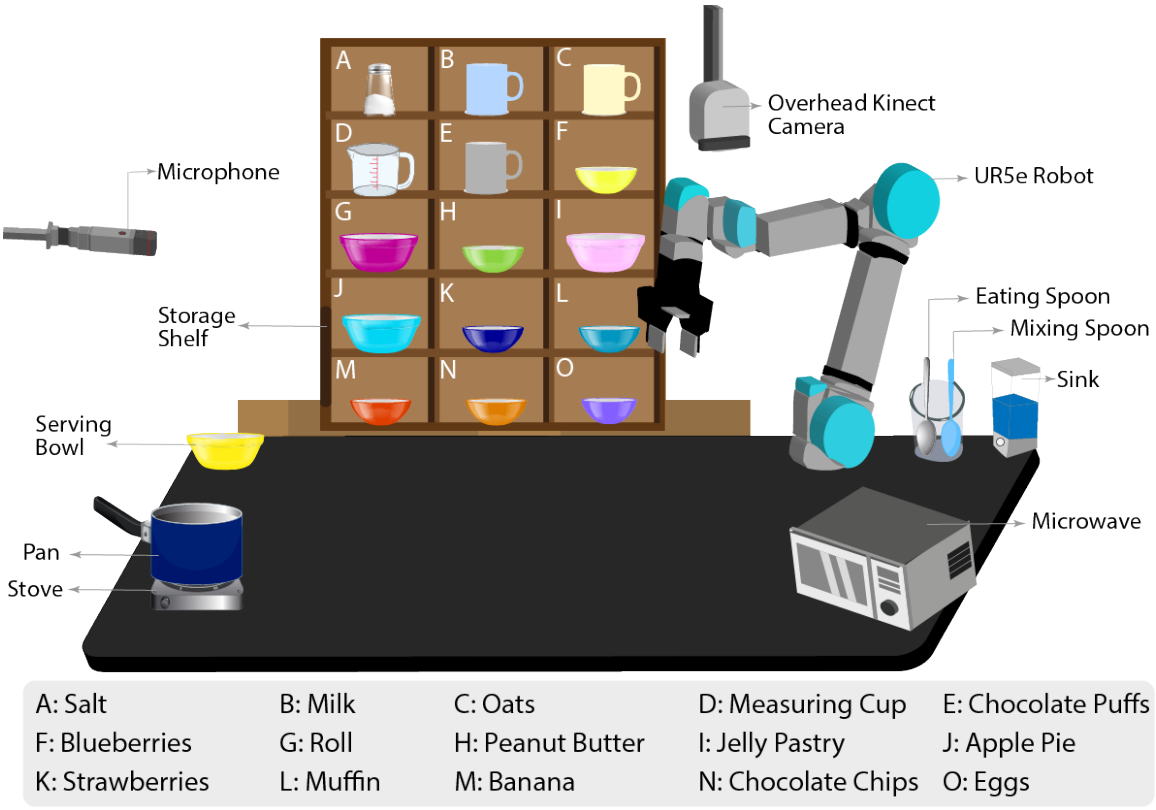


Figure 7.2: Our experiments were performed in a real and simulated kitchen environments. In our real kitchen environment, depicted here, we utilized artificial ingredients and appliances to ensure equipment safety.

generalization capabilities to new preferences than the classic tabular approach. The robot can take actions $A = \{$`gather(i)`, `pour(i,d,t)`, `turn_on(i,t)`, `mix(i, t)`, `collect_water(t)`, `reduce_heat(i,t)`, `cook_oatmeal_wait(i)`, `take_out_microwave(i,t)`, `put_in_microwave(i,t)`, `microwave_wait(i)`, `serve_oatmeal(i)`$\}$, where i is an item available in the interaction, d is a container such as a bowl or a pan, and $t \in \{$`SAY, DO`$\}$ denotes the support type of the action, except for the `gather` action, which is always performed by the robot. For example, if the robot takes the action `pour(milk, pan, DO)` the robot will physically pour the milk into the pan, whereas the action `pour(milk, pan, SAY)` has the robot verbally prompt the user to do the same. The state space is a Boolean vector where each element corresponds to a feature of the environment, including the presence of an

item in the workspace or storage area; whether an item is in a container or appliance; the power state of the stove and microwave; the state of an item (warmed, boiling). Crucially, the state space also contains a representation of the goal meal. This includes three Boolean vectors relating to the current, high level goal for each dish (i.e. pastry, oatmeal, and cereal) but also dish variants (e.g., plain oatmeal, jelly pastry, etc.). We assume a deterministic environment and a fully observable state space.

7.4.2 Experimental Setup

The goal of the physical and the simulated collaborations is for a user and a robot to prepare a breakfast meal comprised of a main dish (one of six variants of oatmeal or a bowl of cereal) and a side dish (one of five microwaveable food items) per the person's meal preferences. Below we describe our experimental setup for the physical robot and simulation experiments.

Simulations

In the simulation, human meal preferences are represented via ground-truth sequences of desired robot actions derived from corresponding Clique/Chain Hierarchical Task Networks (CC-HTNs; Hayes and Scassellati, 2016), which can model aspects of the tasks that must be done sequentially (e.g., the muffin must be placed in the microwave before the microwave is turned on) or are unordered (e.g., the oatmeal toppings can be retrieved in any order). Deviations from these sequences by the simulated robot result in the user providing low-level directives in the form of corrective actions and negative rewards. Each CC-HTN is parameterized by the desired main and side dishes, the order in which they are to be completed (e.g., the muffin should be prepared before the bowl of cereal), the type of support the robot should provide for each dish (e.g., the robot should physically assist in making cereal, but should only provide verbal guidance while the person prepares the muffin), and the liquid base for the oatmeal

(water or milk), if oatmeal is the desired main course, producing 540 possible meal preferences.

Our primary point of comparison is an adapted version of the action-shielding approach developed in van Waveren et al. (van Waveren et al., 2022) for a robot-assisted cooking task. The authors define and utilize three types of shields: *forbidden action* shields, which block actions from being considered by the policy if they lead to an undesired state, *action refinement* shields, which fall back to a pre-defined sequence of actions if a policy's chosen action lead to an undesirable state, and *alternative item* shields, which re-parameterizes a policy action with a new item. Given that these analyses are concerned with methods for modifying robot policies whilst minimizing low-level user guidance, we omit the use of action refinement shields in our simulations. Action refinement shields can trivialize the results because they always suggest the complete desired action sequence for the task. In addition, we extend the alternative item shield (henceforth *"alternative shield"*) to allow for the re-parameterization of any action parameters, not just those related to items.

We perform two sets of analyses in simulation. In both cases, we first train two variants of a base robot policy on randomly sampled meal CC-HTNs sets with no overlays or shields. One variant contains a representation of the goal meal in the state representation, while the other does not. Then, we sample new sets of CC-HTNs representing new user preferences and for each variant compare:

1. the initial performance of the base policy on these new meal preferences with and without modifications by overlays and shields.

2. how the performance changes with the addition of overlays and shields when the robot trains on these new preferences.

Overlay Generation: At the outset of each testing meal, meal-dependent sets of overlays and shields were automatically generated. The overlay set contained three

overlays pertaining to:

- the main dish and its desired order of completion, e.g., the permissive overlay $\texttt{no\_completed\_dish(s)} \rightarrow \texttt{making\_cereal(a)}$,
- the side dish and its desired order, e.g., the permissive overlay $\texttt{one\_completed\_dish(s)} \rightarrow \texttt{making\_pastry(a)} \wedge \texttt{sweet(a)}$,
- the desired support type from the robot for a meal, e.g., the transfer overlay $\neg\texttt{two\_completed\_meals(s)} \rightarrow \texttt{alternative(a1, a2)} \wedge (\texttt{making\_cereal(a1)} \wedge \texttt{do(a1)}) \vee (\texttt{making\_pastry(a1)} \wedge \texttt{say(a1)})$.

Shield Generation: The shield set was generated to maintain as close to functional parity with the overlay set as possible, though the size of this set was meal-dependent. The shield list always included a forbid shield that encoded the main and side dish and the order in which they were to be completed. The forbid shield prevented the robot from retrieving ingredients from the second meal before the first was completed, and vice versa. Additionally, there were always four alternate item shields mapping each gather action for the undesired sides to the gather action for the desired sides. Finally, a variable number of alternate item shields were used to re-parameterize the support type for the main dish and side dish actions depending on the desired support type for each.

Physical Robot

As shown in Figure 7.2, our physical setup consisted of a UR-5e robot, an overhead Microsoft Azure Kinect camera (Smisek et al., 2013) that monitored the position of ingredients in the workspace, and a microphone for the human collaborator to instruct the robot by providing overlays and corrective actions. We developed a simple language model that mapped templated commands to particular overlay rules (see Table 7.1 for examples). Realistic-looking artificial ingredients were used to

ensure equipment safety, and liquids were replaced with colored beads. Similarly, all appliances were modified to turn on but not emit heat. The storage shelf containing the ingredients required to make the meals was behind the robot. Each ingredient was placed in a distinct colored container, which the robot could bring to the workspace whenever needed. We then used color masking techniques to determine the positions of different ingredients in the workspace, so they could be manipulated as required. The workspace in front of the robot contained a microwave and a pan placed on a stove, both modified to enable the robot to manipulate them easily. To the robot's left, the water-dispensing sink and a stand containing the serving and the eating spoon were placed, and to the robot's right, the serving bowl was located.

7.4.3 Reasoning and Learning

Below we describe the reward administration process as well as the predicates utilized by the overlays.

Rewards

During the collaboration, the robot receives rewards potentially accompanied by a corrective action to be performed the next time-step if the agent performs an incorrect action. In our simulated experiments (see Sec. 7.5.1), these signals are used to update the robot's policy. The robot receives:

- $r = -0.5$, if it performs a completely incorrect action,
- $r = -0.25$, if the action only differs by its support type (e.g., the robot does `collect_water(SAY)` instead of `collect_water(DO)`,
- $r = -1.0$, and the episode is terminated if a robot's action leads to some unrecoverable terminal state (e.g., milk and water are both added to the oatmeal),
- $r = 0.5$, when one of the two dishes is completed,

Directive	Rule
First let's make the pastry, then let' make oatmeal	$\neg$two_completed_meals(s) $\rightarrow$ (making_oatmeal(a) $\wedge$ state(one_completed_dish)) $\vee$ (making_pastry(a) $\wedge$ state(no_completed_dish))
Let's make something fruity!	$\neg$two_completed_meals(s) $\rightarrow$ (making_fruity(a) $\vee$ fruity(a))
Don't use anything with dairy!	$\neg$two_completed_meals(s) $\rightarrow$ $\neg$dairy(a)
You make the rest of the oatmeal!	$\neg$two_completed_meals(s) $\rightarrow$ do(a1) $\wedge$ alternative(a1,a2) $\wedge$ making_oatmeal(a1)
First let's make oatmeal, then let' make a pastry	$\neg$two_completed_meals(s) $\rightarrow$ (making_pastry(a) $\wedge$ state(one_completed_dish)) $\vee$ (making_oatmeal(a) $\wedge$state(no_completed_dish))
First let's make the pastry, then let' make cereal	$\neg$two_completed_meals(s) $\rightarrow$ (making_cereal(a) $\wedge$ state(one_completed_dish)) $\vee$ (making_pastry(a) $\wedge$ state(no_completed_dish))
You make the cereal!	$\neg$two_completed_meals(s) $\rightarrow$ do(a1) $\wedge$ alternative(a1,a2) $\wedge$ making_cereal(a1)
Let's make cereal first, then the pastry	$\neg$two_completed_meals(s) $\rightarrow$ (making_pastry(a) $\wedge$ state(one_completed_dish)) $\vee$ (making_cereal(a) $\wedge$ state(no_completed_dish))
Let's make something with bread!	$\neg$two_completed_meals(s) $\rightarrow$ (making_bread(a)$\vee$ bread(a))

Table 7.1: Example ser-provided directives and their logical rule equivalents

- $r = 1.0$, when the meal is fully complete

Predicates

We define a number of state and action predicates used to construct the overlay rules.

- The dietary preferences are $C_{diet} = \{$`sweet(a)`, `fruity(a)`, `dairy(a)`, `gluten(a)`, `non_vegan(a)`, `protein(a)`$\}$ which return true if a manipulates an ingredient corresponding to the dietary restriction predicate (e.g., the actions `pour(bananas,pan,DO)` and $a =$ `gather(bananas)` both cause `sweet(a)` to evaluate to `True`). For each dietary category, and for each possible dish type, there is a set of predicates C_{making} which return true if an action contributes to the completion of a dish with the desired property (e.g., `making_fruity(a)` and `making_nonvegan(a)` both evaluate to `True` for the `gather(salt)` action as this is a necessary step in making fruity and non-vegan oatmeal variants, respectively.

- $C_{inter} = \{$`do(a)`, `say(a)`, `alternative(a1,a2)`$\}$ are support predicates that track whether an action is a do type or say type or compares whether two actions are equivalent save for their support type, respectively.

- $C_{state} = \{$`one_completed_dish(s)`, `two_completed_dish(s)`, `no_completed_dish(s)`$\}$ track the number of completed dishes.

These predicates, along with the logical operators $\{\wedge, \vee, \neg, \rightarrow\}$ are used to construct overlay rules, which are then evaluated using the logical programming language Prolog (Wielemaker et al., 2012).

7.5 Results

Below we present the results from our simulated experiments and proof-of-concept experiments with a physical robot.

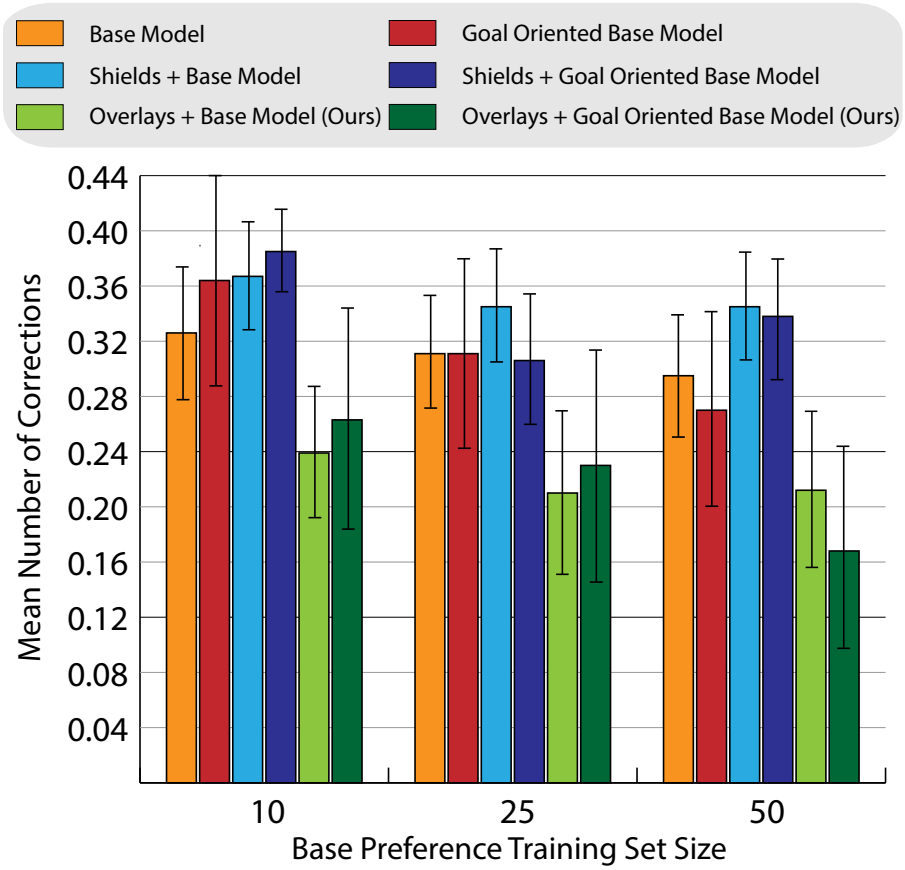


Figure 7.3: We present results comparing adaptation to new meal preferences based on initial preference training set size across model types. The overlay-equipped models generally outperform the other tested models across training set sizes, supporting the notion that overlays can rapidly and accurately adapt a policy.

7.5.1 Simulated Experiments

In our first set of experiments (Figure 7.3) we compare two sets of three models on their ability to adapt to novel meal preferences without additional training. One set of models is trained with an explicit representation of the goal meal in its state space (e.g., oatmeal topped with fruit and a muffin), while the other set was not. Each set trained three base policies on 10, 25, and 50 meals for 50 epochs each, representing three users that differ in the diversity of their meal preferences. We then compared the models' ability to generalize to 50 new meals, including when aided by overlays and shields. The results are averaged over three different runs with three different random

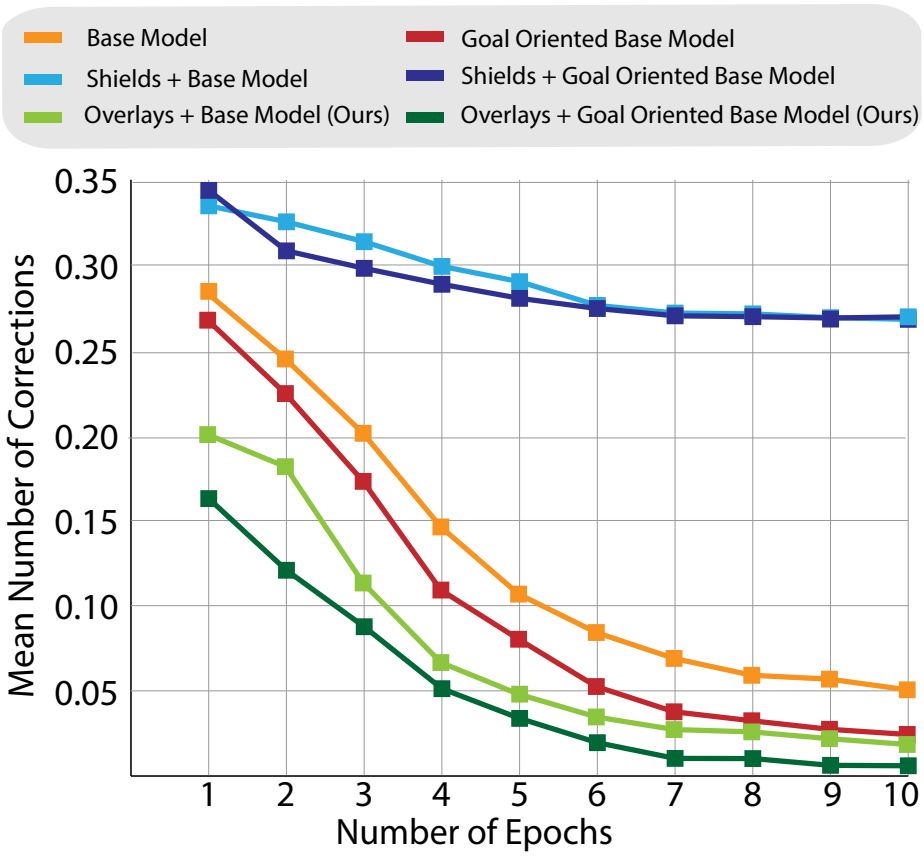


Figure 7.4: We present results comparing the ability of various models to learn new meal preferences. These results support the findings presented in Figure 7.3 suggesting overlay-equipped models' ability to enable immediate adaptations to new preferences. That these performance gains persist across training epochs suggest that overlays can aid in the learning of new preferences as well.

seeds. While there were no significant differences in performance policies trained in a goal-oriented fashion or not, we observed that models trained with overlays typically outperform the base model and the model augmented with shields. This suggests that overlays enabled the robot to exploit aspects of its learning better than the base model can on its own.

For our second set of experiments (Figure 7.4), we explored how overlays and shields benefited the learning of new meal preferences over time. Again, we trained two sets of three base models in the manner described above, each with a different random seed on 50 randomly sampled training meals for 500 episodes and sampled 50

more meals to retrain the policy across 10 episodes. We averaged the results across three runs. While both the base and overlay models improve substantially across the 10 episodes, the overlay model improves more rapidly. As in the previous experiment (see Figure 7.3), the shield model performs the most poorly and notably does not improve substantially despite repeated exposure to the same meal preferences across the 10 episodes.

7.5.2 Physical Robot Experiments

We show three proof-of-concept experiments using a real robot and a human collaborator (here, the researchers taking on the role of a collaborator). These experiments differ from our simulated experiments in two important ways: 1) overlays are added or removed throughout the interaction, rather than just at the interaction's outset, capturing the volatile nature of user preferences; 2) goals are high-level rather than specific (refer to Sec. 7.4.1), modeling user uncertainty regarding the course of the collaboration. Thus, in these physical robot experiments, summarized in Figures 7.5, 7.6a and, 7.6b, the human collaborator adds and/or revokes overlays throughout the collaboration. We also demonstrate instances of the user altering the goal in the middle of a collaboration (Figure 7.6a). For all experiments, we used the same goal-conditioned base policy trained on 50 randomly sampled synthetic meal collaborations (see Sec. 7.4.2) for 500 episodes, the observed convergence point of the learning.

Figure 7.5 summarizes the first experiment. Here the user begins only with the general desire to make oatmeal and a pastry, communicated with the directive *"Let's make the pastry first, and then let's make the oatmeal"* which generates the corresponding permissive overlay and programs the corresponding high-level goal. Subsequently, the user issues three more overlays including one permissive overlay (*"let's make something fruity!"*) one forbid overlay (*"don't use anything with dairy."*), and one transfer overlay (*"You make the rest of the oatmeal"*). As a result, the overlay-

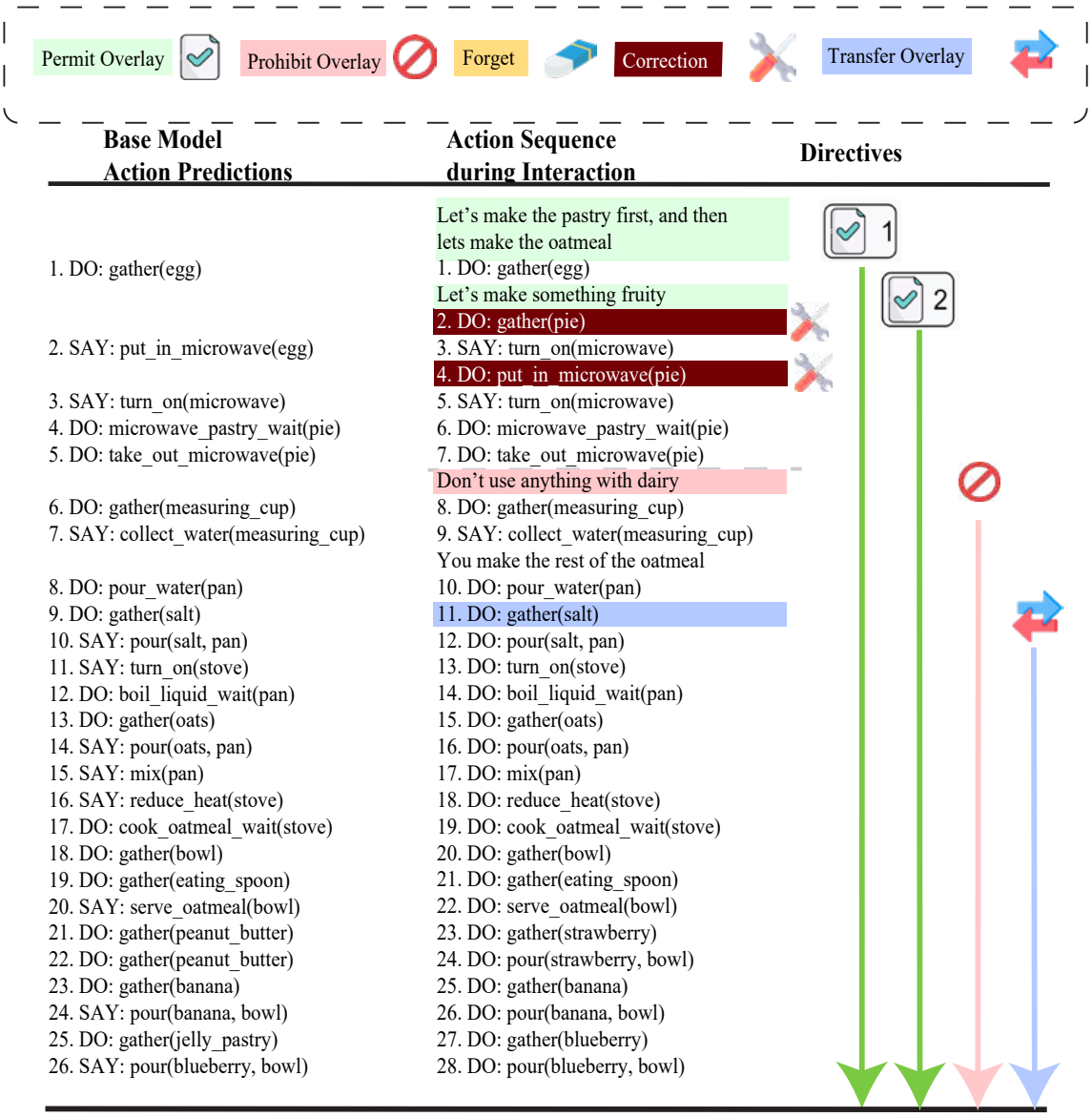


Figure 7.5: We present the first of our three experiments performed with the physical robot utilizing overlays: The first column depicts the action sequence predicted by the base model. The second column shows the action sequence of the policy modified by the overlays and corrective actions, and the third column graphically depicts the applied overlay's activation intervals.

assisted robot makes two errors, while the base model predicted 14 incorrect actions, 6 of which were completely incorrect, while the remaining 8 had the incorrect support type.

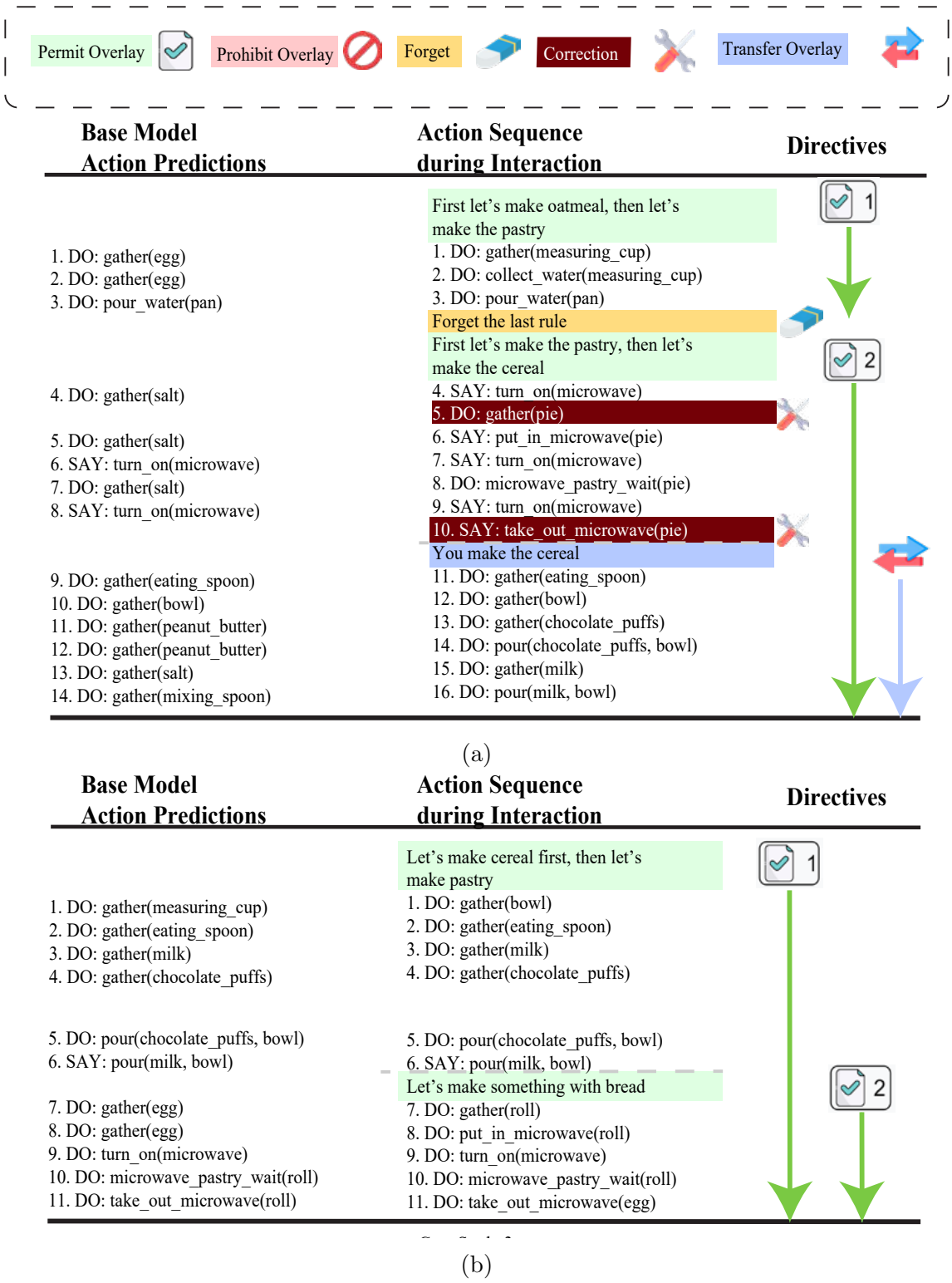


Figure 7.6: We present the results of our second (a) and third (b) robot experiments.

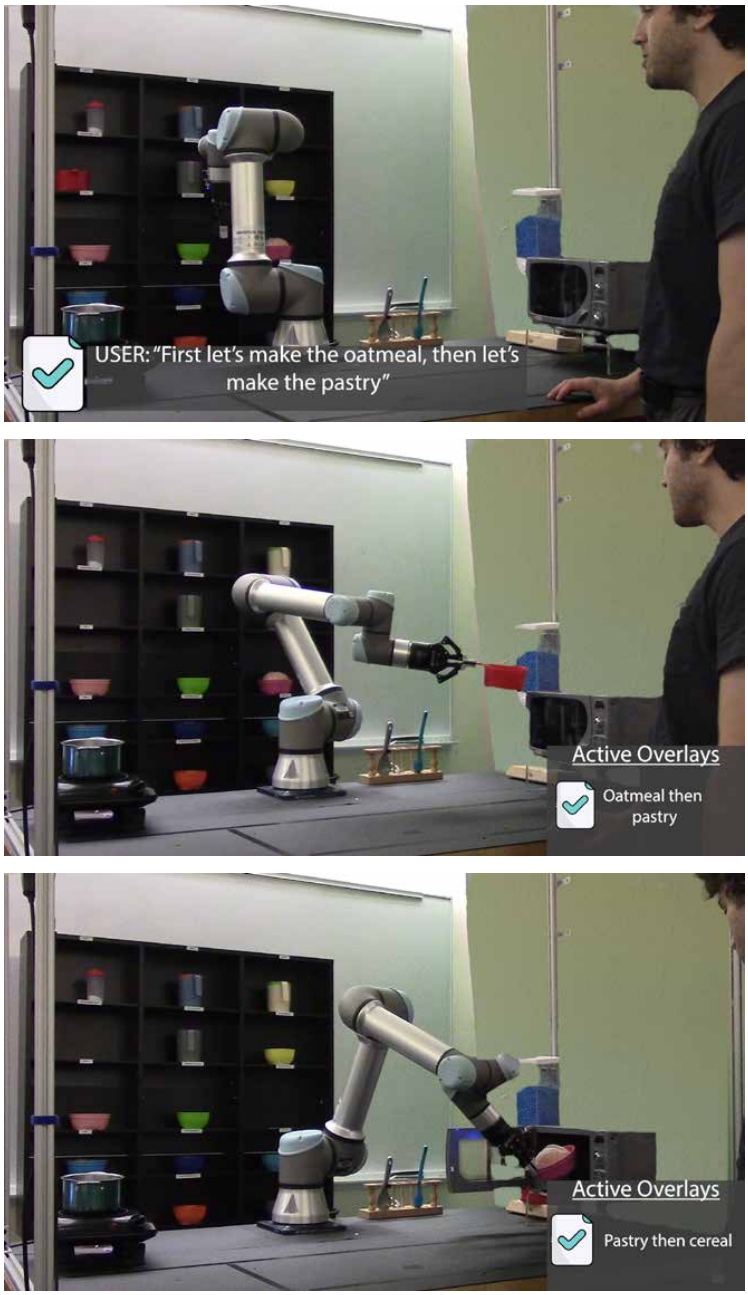


Figure 7.7: Here we show three example interactions from experiment 2. *Top*: At the outset of the interaction, the user provides an overlay, prompting the robot to make oatmeal. Middle: this causes the robot to retrieve water from the sink, a necessary step for oatmeal. *Bottom:* Later on in the interaction, with the user having revoked the original overlay and applied a new one, the robot halts oatmeal preparations and instead prepares a pastry.

Figure 7.6a summarizes the second experiment and Figure 7.7 depicts selected steps from the actual interaction. The user initially prompts the robot with the directive *"first let's make oatmeal, then a pastry,"* communicating both a meal and order preference, as well as conditions the robot's policy on a high-level goal. Of note here is that this overlay is subsequently revoked (*"forget the last rule"*), and replaced with a new permissive overlay and goal (*"let's make the pastry first then let's make the cereal."*) Additionally, the user provides a transfer overlay with the directive *"you make the cereal,"*. In total, the robot makes only two mistakes, while the base model makes 6 incorrect predictions.

Figure 7.6b summarizes the final experiment. First, the user issues the directive *'let's make cereal first, then the pastry,"* expressing a preference for the two dishes and their preferred completion order, and conditions the robot's policy on the high-level meal goal. The user subsequently provides and additional permissive overlay with directive *"let's make something with bread."* In total, the robot makes no errors compared to the base model's 3 prediction errors.

7.6 Discussion

In this chapter, we presented our Transparent Matrix Overlay system and provide validation for this approach with our experiments in a collaborative cooking task on a real robot and in simulation. With these experiments, we sought to answer four questions:

1. *Can overlays aid in immediate adaptation to user preferences?*

   We showed that our approach is able to generalize more effectively to new user preferences by reducing the number of corrective commands a user must issue to the robot in order for it to act according to these preferences. This was true both when overlays were provided at the beginning of a collaboration or added

or revoked within a collaboration, as was presented in our physical robot case studies.

2. *Can overlays be used to aid in the learning of new preferences?*

   Our overlay approach not only improved performance most rapidly when training the policy on novel meal preferences, but it also achieved the lowest error rate after 10 episodes compared to the shielded model and the base policy. This was true for both variants of the base policy equipped with overlays.

3. *How well can overlay-equipped policy adapt to new preferences even when no goals are provided?*

   We did not observe significant differences based on the provision of goals during training. This suggests that overlays can readily produce goal-directed behavior, even when the robot's model possesses no explicit representation of the goal. However, when the model had access to a greater diversity of training meals, we observed the most dramatic improvement in performance when overlays were applied to the goal-oriented base model compared to all other models (refer to Figure 7.4), suggesting that overlay-equipped models have the potential to leverage this information most effectively.

4. *How do overlays compare to other methods regarding (1-3) ?*

   Surprisingly, our adapted version of the shielded approach from (van Waveren et al., 2022) consistently under-performed w.r.t. the baseline policy despite explicitly encoding information about the updated preferences. Figure 7.6a may provide some clues for why this is the case. In certain instances, the base model became biased towards particular incorrect actions (e.g., steps $1-2$, $4-5$). While the overlays would generally suppress such actions from occurring, shields like the alternative item shield utilized in our simulations merely re-parameterized them if applicable, meaning similar biases would continue to

persist. Compounding this issue, shields like the alternate shield mask these biases and prevent them from being easily learned away. That is, alternate shields exchange an action a for another a', but in their shielded training approach, only a' is negatively reinforced. This explains the poor learning gains made by the shielded model as depicted in Figure 7.4, as the model fails to learn to avoid the actions generating these incorrect a's.

However, our approach is not without its limitations. First, we assumed the existence of hand-crafted predicates and associated classifiers rather than learning them autonomously. While we argue that this is a reasonable assumption given the relatively controlled nature of many HRC domains, such assumptions limit our approach's flexibility. Future work should incorporate autonomous methods for state and action abstraction using methods such as those described in Konidaris et al. (2018) and Ugur and Piater (2015). Similarly, other simplifying assumptions were made, such as the existence of a relatively simple, discrete low-level state space, deterministic actions, and non-parallel task completion. Future work should evaluate the efficacy of our approach under more realistic settings.

7.7 Summary

Up until this chapter, we have presented various hybrid approaches tailored to particular aspects of HRC. By combining a predicate-based symbolic interface with RL-learned policies, the method in this chapter represents our initial work toward developing a generalized hybrid approach applicable across HRC. However, as we intend this work to act as a foundation upon which to build, it is not without its limitations. Namely, we focus here only on the top-down modification of a robot policy via a symbolic interface. We do not explore the bottom-up learning of symbols or overlays, a necessary component for making such an approach feasible in real-world applications.

In the next chapter, we will discuss this and other avenues of future research, as well as summarize the contributions of the work in this thesis.

Chapter 8

Conclusion

The integration of symbolic and subsymbolic knowledge has the potential to greatly enhance the fluency, flexibility, and effectiveness of human-robot collaboration (HRC). Our research demonstrates that utilizing both forms of knowledge in concert can not only enhance a robot's ability to learn, reason, and interact with its environment, but also facilitates more natural and intuitive interactions between users and robots. However, the work presented in this thesis is a first step, and further research is needed to improve the robustness and generalizability of these methods for broader real-world applications.

8.1 Contributions

This dissertation makes the following contributions:

- We compile a taxonomy characterizing robot-relevant combined symbolic-subsymbolic approaches (Chapter 2).
- We present a bottom-up hybrid approach for learning and utilizing causal relations for tool affordance reasoning (Chapter 3).

- We present TRI-STAR, a top-down hybrid framework that leverages causal-based task knowledge in order to learn flexible and generalizable tool-use skills (Chapter 4). TRI-STAR enables three capabilities:
  - The ability to learn a tool-use skill from a handful of user demonstrations.
  - The ability to generalize the tool use skills to completely novel tools and manipulated objects without additional training.
  - The transfer of tool-use skills to different robots without additional training.
- We demonstrate how leveraging high level task knowledge in a top-down manner can also improve natural language understanding. In particular, we demonstrate a method by which learned task representations can be used to interpret even highly ambiguous user commands in the context of HRC manufacturing tasks (Chapter 5).
- We present our initial work towards bi-directional symbolic and subsymbolic hybrids, though in the specific context of ownership norm learning (Chapter 6). We devise an approach for the bottom-up, interactive learning of norms relating to object ownership, combined with the top-down inference of ownership relations using these learned norms. We demonstrate the efficacy of this approach in a collaborative task.
- We present the Transparent Matrix Overlay framework, our initial work towards a generalized top-down framework for HRC (Chapter 7). Our framework represents user-specified directives as logical rules that act as constraints on a robot's policy, enabling a user to quickly modify a robot's behavior without the need to retrain it.

8.2 Future Work

The goal of HRC research is to develop robots that can collaborate as effectively as humans do. At a minimum, to achieve this goal, we propose that HRC systems must have the following capabilities: 1) the ability to understand and act on natural language commands and directives from users, 2) the ability to rapidly acquire and adapt to new skills, and 3) the ability to transparently communicate their knowledge and decision-making processes. The research presented in this thesis represents a significant step towards achieving these capabilities, however, further research is needed to fully realize the potential of HRC systems.

8.2.1 Towards the Generalized Integration of User Commands and Directives

As humans tend to naturally communicate in ways that are highly contextual and situated (Shah and Breazeal, 2010), it is important that collaborative robots be capable of this as well. This means not only being able to accurately interpret potentially ambiguous commands but also being able to then integrate those commands directly into a robot's policy. While the work in this thesis tackles both the former (see Chapter 5) and the latter (see Chapters 6 and 7) in isolation, none do so in tandem. This is because in the work described in Chapters 6 and 7, speech commands were translated into corresponding logical rules. As the rules were composed of predicates and logical operators, adapting the NLU system described in Chapter 5 to this context is non-trivial. In practice, this meant designing a templated speech-to-text system that accepted a relatively small number of user commands. However, hand-designing such templates for every conceivable application and potential utterance is infeasible for real-world use cases. One concrete area of future work that could address this would be utilizing large language models like PaLM (Chowdhery et al., 2022) or GTP-3

(Floridi and Chiriatti, 2020) in tandem with our approaches. Such models have not only demonstrated remarkable feats of natural language understanding and generation but are capable of task-agnostic few-shot generalization as well (Brown et al., 2020). One could imagine using such models for producing overlays, for example, by training these models with a handful of examples of natural language commands and their associated logical rule groundings. Such an approach, integrated with a context model such as one described in Chapter 5 could significantly improve the naturalness and fluency by which human collaborators interact with robots, while still maintaining the advantages that overlays provide.

8.2.2 Towards Flexible and Rapid Skill Learning

Construction and manufacturing represent important application domains for HRC (e.g., Roncone et al., 2017; Hayes and Scassellati, 2015; Tellex et al., 2014b). As such, we believe our tool-use framework TRI-STAR (Chapter 4) to be an important step towards the end of designing flexible and rapid skill learners for HRC. However, TRI-STAR was not without its limitations. We made a number of simplifying assumptions in our work that in some cases limit its real-life applicability. For example, TRI-STAR currently does not consider dynamics in its representation of tool-use skills. This is an important omission because in real-world scenarios, tools, and objects may behave differently under different conditions and forces. This limitation could result in the robot's tool-use skills being less robust and adaptable in dynamic environments, making it less effective in human-robot collaboration.

Additionally, there are opportunities to reduce TRI-STAR's reliance on pre-specified knowledge, which also limits its real-world applicability. TRI-STAR's learning capabilities are driven by a pre-specified taxonomic affordance model comprised of cause-and-effect relationships that comprise a class of tool-use tasks. If TRI-STAR were capable of learning causal relations on its own, by, for example, using the causal learn-

ing approach described in Chapter 3, it could enable a robot to update and amend this taxonomy on its own. This, in turn, could potentially enable it to autonomously expand the types of tool-use skills it can acquire, and the scope of collaborative tasks it can accomplish.

8.2.3 Towards Transparent Communication

Many of the approaches described in this thesis learn or utilize structured symbolic representations. The culmination of these works is the Transparent Matrix Overlay system described in Chapter 7, which enables users to modify subsymbolic robot policies via a symbolic rule-based interface. As this approach integrates symbolic knowledge directly into the robot's decision-making process, we believe it to be a tangible step towards transparent HRC, though more work needs to be done. In particular, while symbolic behavioral rules may be more interpretable than a policy instantiated entirely via deep neural networks, they fall short of policy explanations expressed in natural language. One way to remedy this would be to utilize an approach like the one described in Hayes and Shah (2017b) in tandem with overlays. In this approach, policies learned via reinforcement learning are autonomously rendered transparent via a queryable dialogue system. As this system primarily leverages symbolic predicates of the sort utilized by our overlay approach, they are naturally paired. Alternatively, LLMs could potentially be used to transform natural language commands into corresponding overlay rules (refer to Sec. 8.2.1 above), they could also potentially be prompted to provide natural language explanations for active overlays if queried.

Another factor impacting the transparency of the overlay approach is its inability to learn overlay rules. That is, the overlay system acquires overlays only from user directives, and not by, for example, extrapolating rules from its own policy. In theory, though an inductive rule learning approach like the one developed in Chapter 6 could

enable a robot to provide a more transparent representation of its own policy to the user, which in turn could enable a user to better tailor their provided overlays to produce desired robot behaviors.

Though even with the above capabilities in place, the transparency and real-world viability of the overlay approach (and others such as those described in Chapters 3 and 6) is limited by its inability to ground new symbols. In Chapter 7, we assumed the existence of cooking-relevant symbolic predicates accurately grounded to the robot's workspace. Though in practice, it is always possible these pre-specified predicates will fail to fully explain the features in the environment that contribute to the robot's behaviors. Ideally, a robot should be able to recognize for itself which features are relevant, or more fundamentally, how to transform raw sensory data into meaningful abstractions such as features or symbols . Such capabilities could significantly improve the flexibility of these systems as they would limit the amount of pre-specified knowledge required to make these approaches useable. One way to enable these capabilities in practice would be to integrate our approaches with a representation learning approach such as the one described in Konidaris et al. (2018). Here the authors describe a method by which a robot's learned policy can be represented as STRIPS-style rules (Fikes and Nilsson, 1971). Of note is the fact that not only are the rules constructed autonomously but so too are the symbolic predicates that constitute these rules. As the overlay rules are also constructed from such predicates, these approaches are potentially well-paired. This sort of pairing would not only enable a robot to learn new skills and represent them transparently to a user but in turn, via the interface provided by overlays, enable a user to modify these skills as needed.

Appendix A

Additional Methodological Details For Chapter 7

Here we provide additional details regarding our simulated and physical robot experiments described in Chapter 7.

A.1 Model Parameters

Below are the various parameters used to train our single-layer MLP acting as our base DQN in both the simulated and physical experiment.

Parameters	**Values**
Learning Rate (α)	0.002
Discount factor (γ)	.99
Overlay precondition threshold (τ)	.0
Batch Size	64
Number of episodes	500
Exploration rate	0
Hidden layer size	32

Table A.1: Hyperparameters used in our experiments

A.2 Experiments

Table A.2 outlines the complete list of items used in our physical cooking experiments.

Table A.2: Complete list of items used for our Experiments

Category	Items
Base	Oats, Chocolate Puffs
Toppings	Strawberries, Blueberries, Peanut Butter, Chocolate Chips, Banana, Salt Liquid Water, Milk
Side	Roll, Jelly Pastry, Apple Pie, Muffin, Eggs
Appliances	Stove, Microwave, Sink
Utensils	Pan, Serving Bowl, Measuring Cup, Eating Spoon, Mixing Spoon

A.3 Additional Simulation Results

Here we report mean and standard deviations for our second set of simulated experiments (Fig. 4 in the main text).

Table A.3: Mean number of corrections for learning of new meal preference over time for the base model

Epochs	Base Model	Base Model (Goal Conditioned)
1	0.283 ±0.042	0.267 ±0.0628
2	0.244 ±0.058	0.224 ±0.0706
3	0.201 ±0.049	0.172 ±0.0718
4	0.146 ±0.059	0.109 ±0.0633
5	0.106 ±0.054	0.080 ±0.0566
6	0.084 ±0.049	0.052 ±0.0369
7	0.069 ±0.046	0.037 ±0.0335
8	0.059 ±0.037	0.032 ±0.0257
9	0.056 ±0.037	0.027 ±0.0270
10	0.050 ±0.033	0.024 ±0.0224

Table A.4: Mean number of corrections for learning of new meal preference over time for the base model with shields

Epochs	**Base Model + Shields**	**Base model + Shields (goal conditioned)**
1	0.334 ±0.04	0.343 ±0.032
2	0.325 ±0.03	0.308 ±0.029
3	0.313 ±0.03	0.298 ±0.028
4	0.299 ±0.03	0.288 ±0.031
5	0.290 ±0.03	0.280 ±0.039
6	0.276 ±0.02	0.274 ±0.024
7	0.271 ±0.02	0.270 ±0.027
8	0.271 ±0.02	0.269 ±0.026
9	0.269 ±0.03	0.268 ±0.027
10	0.268 ±0.02	0.269 ±0.027

Table A.5: Mean number of corrections for learning of new meal preference over time for the base model with overlays (our method)

Epochs	**Base Model + Overlays**	**Base model + Overlays (goal conditioned)**
1	0.200 ±0.039	0.162 ±0.080
2	0.181 ±0.041	0.120 ±0.082
3	0.113 ±0.038	0.087 ±0.081
4	0.066 ±0.036	0.051 ±0.077
5	0.048 ±0.032	0.033 ±0.071
6	0.034 ±0.030	0.019 ±0.043
7	0.027 ±0.022	0.010 ±0.015
8	0.025 ±0.025	0.010 ±0.019
9	0.021 ±0.023	0.006 ±0.011
10	0.018 ±0.017	0.006 ±0.011

A.4 Physical Robot Experimental Details

A.4.1 Low-level motion trajectories

We utilized both calculated inverse kinematics trajectories and when possible hard-coded forward-kinematics trajectories. All ingredients behind the robot as shown in Fig. 2 of the main text were brought into the workspace with hard-coded trajectories using forward kinematics. All ingredients in the workspace in front of the robot were manipulated by planning inverse kinematics trajectories. First the ingredient that needed to be manipulated was identified using color masking. Then, the robot planned a trajectory using an inverse kinematics solver to pick up the item. Finally, depending on the final destination of the ingredient, it was placed in the destination location using a hard-coded forward kinematics trajectory.

Bibliography

Abbeel, P., Quigley, M., and Ng, A. Y. (2006). Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8.

Abe, S. (2010). Feature selection and extraction. In *Support Vector Machines for Pattern Classification*, pages 331–341. Springer.

Abelha, P. and Guerin, F. (2017). Learning how a tool affords by simulating 3d models from the web. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4923–4929. IEEE.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR.

Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., and Kaempchen, N. (2015). Experience, results and lessons learned from automated driving on germany's highways. *IEEE Intelligent transportation systems magazine*, 7(1):42–57.

Agrawal, P. (2022). The task specification problem. In *Conference on Robot Learning*, pages 1745–1751. PMLR.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter,

B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. (2022). Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Akrour, R., Schoenauer, M., and Sebag, M. (2011). Preference-based policy learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 12–27. Springer.

Akrour, R., Schoenauer, M., and Sebag, M. (2012). April: Active preference learning-based reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 116–131. Springer.

Alomari, M., Duckworth, P., Hogg, D., and Cohn, A. (2017). Natural language acquisition and grounding for embodied robotic systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U. (2018a). Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U. (2018b). Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Altman, E. (1999). *Constrained Markov decision processes: stochastic modeling*. Routledge.

Anderson, J. R., Matessa, M., and Lebiere, C. (1997). Act-r: A theory of higher

level cognition and its relation to visual attention. *Human–Computer Interaction*, 12(4):439–462.

Andries, M., Chavez-Garcia, R. O., Chatila, R., Giusti, A., and Gambardella, L. M. (2018). Affordance equivalences in robotics: a formalism. *Frontiers in neurorobotics*, 12:26.

Angelov, D., Hristov, Y., and Ramamoorthy, S. (2019). Using causal analysis to learn specifications from task demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1341–1349. International Foundation for Autonomous Agents and Multiagent Systems.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

Arnay, R., Morales, N., Morell, A., Hernandez-Aceituno, J., Perea, D., Toledo, J. T., Hamilton, A., Sanchez-Medina, J. J., and Acosta, L. (2016). Safe and reliable path planning for the autonomous vehicle verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Bader, S. and Hitzler, P. (2005). Dimensions of neural-symbolic integration-a structured survey. *arXiv preprint cs/0511042*.

Bagnell, J. A. and Schneider, J. G. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pages 1615–1620. IEEE.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Berg, M., Bayazit, D., Mathew, R., Rotter-Aboyoun, A., Pavlick, E., and Tellex, S. (2020). Grounding language to landmarks in arbitrary outdoor environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 208–215. IEEE.

Biyik, E. and Sadigh, D. (2018). Batch active preference-based learning of reward functions. In *Conference on robot learning*, pages 519–528. PMLR.

Bougie, N. and Ichise, R. (2020). Towards interpretable reinforcement learning with state abstraction driven by external knowledge. *IEICE Transactions on Information and Systems*, 103(10):2143–2153.

Brandi, S., Kroemer, O., and Peters, J. (2014). Generalizing pouring actions between objects using warped parameters. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 616–621. IEEE.

Brawer, J., Ghose, D., Candon, K., Qin, M., Roncone, A., Vazquez, M., and Scassellati, B. (2023). Interactive policy shaping for human-robot collaboration with transparent matrix overlays. In *Proceedings of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*.

Brawer, J., Mangin, O., Roncone, A., Widder, S., and Scassellati, B. (2018). Situated human–robot collaboration: predicting intent from grounded natural language. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 827–833. IEEE.

Brawer, J., Qin, M., and Scassellati, B. (2020). A causal approach to tool affordance learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8394–8399. IEEE.

Breazeal, C., Hoffman, G., and Lockerd, A. (2004). Teaching and working with robots as a collaboration. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1030–1037. IEEE Computer Society.

Brechtel, S., Gindele, T., and Dillmann, R. (2011). Probabilistic mdp-behavior planning for cars. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1537–1542. IEEE.

Brick, T., Schermerhorn, P., and Scheutz, M. (2007). Speech and action: Integration of action and language for mobile robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1423–1428. IEEE.

Brick, T. and Scheutz, M. (2007). Incremental natural language processing for hri. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 263–270.

Briggs, G. and Scheutz, M. (2011). Facilitating Mental Modeling in Collaborative Human-robot Interaction Through Adverbial Cues. In *Proceedings of the SIGDIAL 2011 Conference*, pages 239–247, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., Mc-

Candlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

Brubaker, M. A., Geiger, A., and Urtasun, R. (2015). Map-based probabilistic visual self-localization. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):652–665.

Büchi, J. R. (1990). On a decision method in restricted second order arithmetic. In *The collected works of J. Richard Büchi*, pages 425–435. Springer.

Bucker, A., Figueredo, L., Haddadin, S., Kapoor, A., Ma, S., and Bonatti, R. (2022). Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers. *arXiv preprint arXiv:2203.13411*.

Burger, B., Maffettone, P. M., Gusev, V. V., Aitchison, C. M., Bai, Y., Wang, X., Li, X., Alston, B. M., Li, B., Clowes, R., et al. (2020). A mobile robotic chemist. *Nature*, 583(7815):237–241.

Bütepage, J. and Kragic, D. (2017). Human-Robot Collaboration: From Psychology to Social Robotics. *CoRR*.

Camacho, A., Chen, O., Sanner, S., and McIlraith, S. A. (2017). Non-markovian rewards expressed in ltl: guiding search via reward shaping. In *Tenth annual symposium on combinatorial search*.

Camacho, A., Icarte, R. T., Klassen, T. Q., Valenzano, R. A., and McIlraith, S. A. (2019). Ltl and beyond: Formal languages for reward function specification in reinforcement learning. In *IJCAI*, volume 19, pages 6065–6073.

Cantrell, R., Benton, J., Talamadupula, K., Kambhampati, S., Schermerhorn, P., and Scheutz, M. (2012). Tell me when and why to do it! Run-time planner model

updates via natural language instruction. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 471–478. IEEE.

Cantrell, R., Schermerhorn, P., and Scheutz, M. (2011). Learning actions from human-robot dialogues. In *RO-MAN, 2011 IEEE*, pages 125–130. IEEE.

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. (2019). On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32.

Chasles, M. (1830). *Mémoire de géométrie pure sur les systèmes de forces, et les systèmes d'aires planes; et sur les polygones, les polyèdres, et les centres des moyennes distances.(Extrait du VIe volume de la Correspondence mathématique et physique des Pays-Bas.).* Mayez.

Chen, C., Li, H.-X., and Dong, D. (2008). Hybrid control for robot navigation-a hierarchical q-learning algorithm. *IEEE Robotics & Automation Magazine*, 15(2):37–47.

Chitnis, R., Silver, T., Tenenbaum, J. B., Kaelbling, L. P., and Lozano-Pérez, T. (2021). Glib: Efficient exploration for relational model-based reinforcement learning via goal-literal babbling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11782–11791.

Chong, H.-Q., Tan, A.-H., and Ng, G.-W. (2007). Integrated cognitive architectures: a survey. *Artificial Intelligence Review*, 28(2):103–130.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017).

Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Clark, P. and Boswell, R. (1991). Rule induction with cn2: Some recent improvements. In *European Working Session on Learning*, pages 151–163. Springer.

Coppens, Y., Efthymiadis, K., Lenaerts, T., Nowé, A., Miller, T., Weber, R., and Magazzeni, D. (2019). Distilling deep reinforcement learning policies in soft decision trees. In *Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence*, pages 1–6.

Corapi, D., Russo, A., De Vos, M., Padget, J., and Satoh, K. (2011). Normative design using inductive learning. *Theory and Practice of Logic Programming*, 11(4-5):783–799.

Cornia, M., Baraldi, L., and Cucchiara, R. (2020). Smart: training shallow memory-aware transformers for robotic explainability. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1128–1134. IEEE.

Cranefield, S., Savarimuthu, T., Meneguzzi, F., and Oren, N. (2015). A bayesian approach to norm identification. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1743–1744. International Foundation for Autonomous Agents and Multiagent Systems.

Cui, Y., Zhang, Q., Allievi, A., Stone, P., Niekum, S., and Knox, W. B. (2020). The empathic framework for task learning from implicit human feedback. *arXiv preprint arXiv:2009.13649*.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Daniel, W. W. (1990). Kolmogorov–smirnov one-sample test. *Applied nonparametric statistics*, 2.

De Campos, C. P., Zeng, Z., and Ji, Q. (2009). Structure learning of bayesian networks using constraints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 113–120.

De Raedt, L., Dries, A., Thon, I., Van den Broeck, G., and Verbeke, M. (2015). Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of 24th international joint conference on artificial intelligence (IJCAI)*, pages 1835–1842.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007). Problog: a probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, pages 2468–2473. Morgan Kaufmann Publishers Inc.

De Raedt, L. and Thon, I. (2010). Probabilistic rule learning. In *ILP*, pages 47–58. Springer.

Dehban, A., Jamone, L., Kampff, A. R., and Santos-Victor, J. (2016). Denoising auto-encoders for learning of objects and tools affordances in continuous space. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4866–4871. IEEE.

Deisenroth, M. P., Englert, P., Peters, J., and Fox, D. (2014). Multi-task policy search for robotics. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 3876–3881. IEEE.

Deits, R., Tellex, S., Thaker, P., Simeonov, D., Kollar, T., and Roy, N. (2013). Clarifying commands with information-theoretic human-robot dialog. *Journal of Human-Robot Interaction*, 2(2):58–79.

Dennis, Jr, J. E. and Moré, J. J. (1977). Quasi-newton methods, motivation and theory. *SIAM review*, 19(1):46–89.

Detry, R., Papon, J., and Matthies, L. (2017). Task-oriented grasping with semantic and geometric scene understanding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3266–3273. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303.

Dijkstra, E. W. (2022). A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290.

Dindo, H. and Zambuto, D. (2010). A probabilistic approach to learning a visually grounded language model through human-robot interaction. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 790–796. IEEE.

Dreyfus, H., Dreyfus, S. E., and Athanasiou, T. (2000). *Mind over machine*. Simon and Schuster.

Driessens, K. and Ramon, J. (2003). Relational instance based regression for relational reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 123–130.

Droniou, A., Ivaldi, S., and Sigaud, O. (2014). Learning a repertoire of actions with deep neural networks. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 229–234. IEEE.

Drton, M. and Maathuis, M. H. (2017). Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393.

Eaton, D. and Murphy, K. (2012). Bayesian structure learning using dynamic programming and mcmc. *arXiv preprint arXiv:1206.5247*.

Eberhardt, F. (2017). Introduction to the foundations of causal discovery. *International Journal of Data Science and Analytics*, 3(2):81–91.

Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V., and Uras, T. (2011). Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 4575–4581. IEEE.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd*, 96(34):226–231.

Fang, K., Zhu, Y., Garg, A., Kurenkov, A., Mehta, V., Fei-Fei, L., and Savarese, S. (2020). Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216.

Fikes, R. E. and Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fitzgerald, T., Short, E., Goel, A., and Thomaz, A. (2019). Human-guided trajectory adaptation for tool transfer. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1350–1358.

Floridi, L. and Chiriatti, M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30(4):681–694.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Forechi, A., Oliveira-Santos, T., Badue, C., and De Souza, A. F. (2018). Visual global localization with a hybrid wnn-cnn approach. In *2018 international joint conference on neural networks (IJCNN)*, pages 1–9. IEEE.

Fransen, B., Morariu, V., Martinson, E., Blisard, S., Marge, M., Thomas, S., Schultz, A., and Perzanowski, D. (2007). Using vision, acoustics, and natural language for disambiguation. In *Proceeding of the ACM/IEEE international conference on Human-robot interaction (HRI)*. ACM Press.

Frasca, T., Oosterveld, B., Chita-Tegmark, M., and Scheutz, M. (2021). Enabling fast instruction-based modification of learned robot skills. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6075–6083.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54.

Gajewski, P., Ferreira, P., Bartels, G., Wang, C., Guerin, F., Indurkhya, B., Beetz, M., and Śniezyński, B. (2019). Adapting everyday manipulation skills to varied scenarios. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1345–1351. IEEE.

Galceran, E., Cunningham, A. G., Eustice, R. M., and Olson, E. (2017). Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment. *Autonomous Robots*, 41(6):1367–1382.

Galindo, C. and Saffiotti, A. (2012). Semantic norms for mobile robots: When the end does not justify the means. In *c*, pages 84–89.

Gao, W. and Tedrake, R. (2021). kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6(2):2962–2969.

Garcez, A. d. and Lamb, L. C. (2020). Neurosymbolic ai: the 3rd wave. *arXiv preprint arXiv:2012.05876*.

Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. (2021a). Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. (2021b). Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293.

Garrett, C. R., Lozano-Perez, T., and Kaelbling, L. P. (2018). Ffrob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 37(1):104–136.

Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2020). Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292.

Gazdar, G. (1980). Pragmatics, Implicature, Presuposition and Lógical Form. *Crítica: Revista Hispanoamericana de Filosofía*, 12(35):113–122.

Ghayoumi, M. and Pourebadi, M. (2019). Fuzzy knowledge-based architecture for learning and interaction in social robots. *arXiv preprint arXiv:1909.11004*.

Giacobbe, M., Hasanbeig, M., Kroening, D., and Wijk, H. (2021). Shielding atari games with bounded prescience. *arXiv preprint arXiv:2101.08153*.

Gibson, J. J. (1979). The ecological approach to visual perception.

Giuliari, F., Skenderi, G., Cristani, M., Wang, Y., and Del Bue, A. (2022). Spatial commonsense graph for object localisation in partial scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19518–19527.

Goldberg, A. V. and Harrelson, C. (2005). Computing the shortest path: A search meets graph theory. In *SODA*, volume 5, pages 156–165.

Gonçalves, A., Abrantes, J., Saponaro, G., Jamone, L., and Bernardino, A. (2014a). Learning intermediate object affordances: Towards the development of a tool concept. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 482–488. IEEE.

Gonçalves, A., Saponaro, G., Jamone, L., and Bernardino, A. (2014b). Learning visual affordances of objects and tools through autonomous robot exploration. In *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 128–133. IEEE.

Gopalan, N., Arumugam, D., Wong, L. L., and Tellex, S. (2018). Sequence-to-sequence language grounding of non-markovian task specifications. In *Robotics: Science and Systems*, volume 2018.

Goudet, O., Kalainathan, D., Caillou, P., Guyon, I., Lopez-Paz, D., and Sebag, M. (2017). Causal generative neural networks. *arXiv preprint arXiv:1711.08936*.

Gravot, F., Cambon, S., and Alami, R. (2005). asymov: a planner that deals with intricate symbolic and geometric problems. In *Robotics Research. The Eleventh International Symposium*, pages 100–110. Springer.

Grice, H. P. (1975). Logic and conversation. In Ezcurdia, M. and Stainton, R. J., editors, *The Semantics-Pragmatics Boundary in Philosophy*, page 47. Broadview Press.

Guadarrama, S., Riano, L., Golland, D., Go, D., Jia, Y., Klein, D., Abbeel, P., Darrell, T., et al. (2013). Grounding spatial relations for human-robot interaction. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1640–1647. IEEE.

Guhur, P.-L., Chen, S., Garcia, R., Tapaswi, M., Laptev, I., and Schmid, C. (2022). Instruction-driven history-aware policies for robotic manipulations. *arXiv preprint arXiv:2209.04899*.

Gupta, U. D., Talvitie, E., and Bowling, M. (2015). Policy tree: Adaptive representation for policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Gutman, R. J. (2004). Reach-based routing: A new approach to shortest path algorithms optimized for road networks. *ALENEX/ANALC*, 4:100–111.

Hanford, S. D., Janrathitikarn, O., and Long, L. N. (2009). Control of mobile robots using the soar cognitive architecture. *Journal of Aerospace Computing, Information, and Communication*, 6(2):69–91.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hasan, M., Warburton, M., Agboh, W. C., Dogar, M. R., Leonetti, M., Wang, H., Mushtaq, F., Mon-Williams, M., and Cohn, A. G. (2020). Human-like planning for reaching in cluttered environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7784–7790. IEEE.

Hata, A. Y. and Wolf, D. F. (2015). Feature detection for vehicle localization in urban environments using a multilayer lidar. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):420–429.

Haugeland, J. (1989). *Artificial intelligence: The very idea*. MIT press.

Hauser, A. and Bühlmann, P. (2014). Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55(4):926–939.

Hayes, B. and Scassellati, B. (2013). Challenges in shared-environment human-robot collaboration. *learning*, 8(9).

Hayes, B. and Scassellati, B. (2015). Effective robot teammate behaviors for supporting sequential manipulation tasks. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*.

Hayes, B. and Scassellati, B. (2016). Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5476. IEEE.

Hayes, B. and Shah, J. A. (2017a). Improving robot controller transparency through

autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE.

Hayes, B. and Shah, J. A. (2017b). Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 303–312. IEEE.

He, M., Takeuchi, E., Ninomiya, Y., and Kato, S. (2016). Precise and efficient model-based vehicle tracking method using rao-blackwellized and scaling series particle filters. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 117–124. IEEE.

Heckerman, D., Meek, C., and Cooper, G. (1997). A bayesian approach to causal discovery. Technical report, Technical report msr-tr-97-05, Microsoft Research.

Hejna, J. and Sadigh, D. (2022). Few-shot preference learning for human-in-the-loop rl. *arXiv preprint arXiv:2212.03363*.

Hemachandra, S., Walter, M. R., Tellex, S., and Teller, S. (2014). Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2623–2630. IEEE.

Hillenbrand, U. and Roa, M. A. (2012). Transferring functional grasps through contact warping and local replanning. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2963–2970. IEEE.

Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266.

Hoekstra, R., Breuker, J., Di Bello, M., Boer, A., et al. (2007). The lkif core ontology of basic legal concepts. *LOAIT*, 321:43–63.

Hoffmann, H., Chen, Z., Earl, D., Mitchell, D., Salemi, B., and Sinapov, J. (2014). Adaptive robotic tool use under variable grasps. *Robotics and Autonomous Systems*, 62(6):833–846.

Hong, J., Mozetic, I., and Michalski, R. S. (1986). Aq15: Incremental learning of attribute-based descriptions from examples: The method and user's guide. Technical report, Department of Computer Science, University of Illinois, Urbana-Champaign.

Hou, Z., Fei, J., Deng, Y., and Xu, J. (2020). Data-efficient hierarchical reinforcement learning for robotic assembly control applications. *IEEE Transactions on Industrial Electronics*, 68(11):11565–11575.

Howard, T. M., Tellex, S., and Roy, N. (2014). A natural language planner interface for mobile manipulators. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6652–6659. IEEE.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2008). Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21.

Hsiung, E., Mehta, H., Chu, J., Liu, X., Patel, R., Tellex, S., and Konidaris, G. (2022a). Generalizing to new domains by mapping natural language to lifted ltl. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3624–3630. IEEE.

Hsiung, E., Mehta, H., Chu, J., Liu, X., Patel, R., Tellex, S., and Konidaris, G. (2022b). Generalizing to new domains by mapping natural language to lifted ltl. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3624–3630.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. (2022a). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022b). Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. (2018). Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31.

Iio, T., Shiomi, M., Shinozawa, K., Miyashita, T., Akimoto, T., and Hagita, N. (2009). Lexical entrainment in human-robot interaction: Can robots entrain human vocabulary? In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3727–3734.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE.

Iucci, A., Hata, A., Terra, A., Inam, R., and Leite, I. (2021). Explainable reinforcement learning for human-robot collaboration. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 927–934. IEEE.

Jansen, N., Könighofer, B., Junges, S., and Bloem, R. (2018a). Shielded decision-making in mdps. *arXiv preprint arXiv:1807.06096*.

Jansen, N., Könighofer, B., Junges, S., Serban, A. C., and Bloem, R. (2018b). Safe reinforcement learning via probabilistic shields. *arXiv preprint arXiv:1807.06096*.

Jelbert, S. A., Taylor, A. H., Cheke, L. G., Clayton, N. S., and Gray, R. D. (2014). Using the aesop's fable paradigm to investigate causal understanding of water displacement by new caledonian crows. *PloS one*, 9(3):e92895.

Jenkins, O. C. and Matarić, M. J. (2004). A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56.

Ji, Y., Li, Z., Sun, Y., Peng, X. B., Levine, S., Berseth, G., and Sreenath, K. (2022). Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. *arXiv preprint arXiv:2208.01160*.

Jo, K., Kim, J., Kim, D., Jang, C., and Sunwoo, M. (2015). Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. *IEEE Transactions on Industrial Electronics*, 62(8):5119–5132.

Johannsmeier, L. and Haddadin, S. (2017). A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes. *IEEE Robotics and Automation Letters*, 2(1):41–48.

Johns, J., Painter-Wakefield, C., and Parr, R. (2010). Linear complementarity for regularized policy evaluation and improvement. *Advances in neural information processing systems*, 23.

Jonschkowski, R. and Brock, O. (2015). Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428.

Kalainathan, D. and Goudet, O. (2019). Causal discovery toolbox: Uncover causal relationships in python. *arXiv preprint arXiv:1903.02278*.

Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., and Agrawal, H. (2022). Housekeep: Tidying virtual households using commonsense reasoning. *arXiv preprint arXiv:2205.10712*.

Kasenberg, D. and Scheutz, M. (2018a). Inverse norm conflict resolution. In *Proceedings of the First AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society*.

Kasenberg, D. and Scheutz, M. (2018b). Norm conflict resolution in stochastic domains. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Kautz, H. (2022). The third ai summer: Aaai robert s. engelmore memorial lecture. *AI Magazine*, 43(1):93–104.

Kawano, H. (2013). Hierarchical sub-task decomposition for reinforcement learning of multi-robot delivery mission. In *2013 IEEE International Conference on Robotics and Automation*, pages 828–835. IEEE.

Kazemitabar, S. J. and Beigy, H. (2008). Automatic discovery of subgoals in reinforcement learning using strongly connected components. In *International Conference on Neural Information Processing*, pages 829–834. Springer.

Kemp, C. C. and Edsinger, A. (2006). Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *Proc. of the Fifth Intl. Conference on Development and Learning*, volume 42.

Kersting, K., Otterlo, M. V., and De Raedt, L. (2004). Bellman goes relational. In *Proceedings of the twenty-first international conference on Machine learning*, page 59.

Knepper, R. A., Layton, T., Romanishin, J., and Rus, D. (2013). IkeaBot: An autonomous multi-robot coordinated furniture assembly system. In *IEEE International Conference on Robotics and Automation*.

Kober, J., Mohler, B., and Peters, J. (2008). Learning perceptual coupling for motor primitives. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 834–839. IEEE.

Kocaoglu, M., Snyder, C., Dimakis, A. G., and Vishwanath, S. (2017). Causalgan: Learning causal implicit generative models with adversarial training. *arXiv preprint arXiv:1709.02023*.

Kokic, M., Stork, J. A., Haustein, J. A., and Kragic, D. (2017). Affordance detection for task-specific grasping using deep learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 91–98. IEEE.

Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Human-Robot Interaction (HRI), 5th ACM/IEEE International Conference on*, pages 259–266. IEEE.

Konidaris, G. (2019). On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289.

Könighofer, B., Rudolf, J., Palmisano, A., Tappler, M., and Bloem, R. (2021). Online shielding for stochastic systems. In *NASA Formal Methods Symposium*, pages 231–248. Springer.

Kroon, M. and Whiteson, S. (2009). Automatic feature selection for model-based reinforcement learning in factored mdps. In *2009 International Conference on Machine Learning and Applications*, pages 324–330. IEEE.

Ku, L. Y., Sen, S., Learned-Miller, E. G., and Grupen, R. A. (2014). The aspect transition graph: An affordance-based model. In *European Conference on Computer Vision*, pages 459–465. Springer.

Kumar, N., McClinton, W., Chitnis, R., Silver, T., Lozano-Pérez, T., and Kaelbling, L. P. (2022). Learning operators with ignore effects for bilevel planning in continuous domains. *arXiv preprint arXiv:2208.07737*.

Kupcsik, A., Deisenroth, M. P., Peters, J., Loh, A. P., Vadakkepat, P., and Neumann, G. (2017). Model-based contextual policy search for data-efficient generalization of robot skills. *Artificial Intelligence*, 247:415–439.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64.

Lam, H.-P., Hashmi, M., and Scofield, B. (2016). Enabling reasoning with legalruleml. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 241–257. Springer.

Landajuela, M., Petersen, B. K., Kim, S., Santiago, C. P., Glatt, R., Mundhenk, N., Pettit, J. F., and Faissol, D. (2021). Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR.

Langley, P. and Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):54–64.

Lee, J. and Moray, N. (1992). Trust, control strategies and allocation of function in human-machine systems. *Ergonomics*, 35(10):1243–1270.

Lee, K., Smith, L., and Abbeel, P. (2021a). Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*.

Lee, K., Smith, L., Dragan, A., and Abbeel, P. (2021b). B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*.

Lee, K.-H., Nachum, O., Yang, M., Lee, L., Freeman, D., Xu, W., Guadarrama, S., Fischer, I., Jang, E., Michalewski, H., et al. (2022). Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*.

Leonard, M., Graham, S., and Bonacum, D. (2004). The human factor: the critical importance of effective teamwork and communication in providing safe care. *BMJ Quality & Safety*, 13(suppl 1):i85–i90.

Levy, A., Platt, R., and Saenko, K. (2019). Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*.

Limeros, S. C., Majchrowska, S., Johnander, J., Petersson, C., and Llorca, D. F. (2022). Towards explainable motion prediction using heterogeneous graph representations. *arXiv preprint arXiv:2212.03806*.

Lioutikov, R., Neumann, G., Maeda, G., and Peters, J. (2017). Learning movement primitive libraries through probabilistic segmentation. *The International Journal of Robotics Research*, 36(8):879–894.

Liu, M., Zhu, M., and Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*.

Loftin, R., Peng, B., MacGlashan, J., Littman, M. L., Taylor, M. E., Huang, J., and Roberts, D. L. (2016). Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous agents and multi-agent systems*, 30(1):30–59.

Lohse, M., Rohlfing, K. J., Wrede, B., and Sagerer, G. (2008). Try something else! When users change their discursive behavior in human-robot interaction. In *2008 IEEE International Conference on Robotics and Automation*, pages 3481–3486.

Lueddecke, T., Kulvicius, T., and Woergoetter, F. (2019). Context-based affordance segmentation from 2d images for robot actions. *Robotics and Autonomous Systems*, 119:92–107.

Lynch, K. M. and Park, F. C. (2017). *Modern robotics*. Cambridge University Press.

Lyu, D., Yang, F., Liu, B., and Gustafson, S. (2019). Sdrl: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2970–2977.

MacGlashan, J., Babes-Vroman, M., desJardins, M., Littman, M., Muresan, S., Squire, S., Tellex, S., Arumugam, D., and Yang, L. (2015). Grounding English Commands to Reward Functions. In *Proceedings of Robotics: Science and Systems*, Rome, Italy.

Magder, L. S. and Hughes, J. P. (1997). Logistic regression when the outcome is measured with uncertainty. *American Journal of Epidemiology*, 146(2):195–203.

Mahadevan, S., Maggioni, M., Ferguson, K., and Osentoski, S. (2006). Learning representation and control in continuous markov decision processes. In *AAAI*, volume 6, pages 1194–1199.

Malle, B. F., Scheutz, M., and Austerweil, J. L. (2017). Networks of social and moral norms in human and robot agents. In *A World with Robots*, pages 3–17. Springer.

Maloof, M. A. (2003). Incremental rule learning with partial instance memory for changing concepts. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2764–2769. IEEE.

Mangin, O., Roncone, A., and Scassellati, B. (2017). How to be helpful? implementing supportive behaviors for human-robot collaboration. *arXiv preprint arXiv:1710.11194*.

Mangin, O., Roncone, A., and Scassellati, B. (2022). How to be helpful? supportive behaviors and personalization for human-robot collaboration. *Frontiers in Robotics and AI*, page 426.

Mannor, S., Menache, I., Hoze, A., and Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 71.

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. (2019). kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer.

Mar, T., Tikhanoff, V., Metta, G., and Natale, L. (2017). Self-supervised learning of tool affordances from 3d tool representation through parallel som mapping. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 894–901. IEEE.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marcus, G. (2020). The next decade in ai: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

Marcus, G., Davis, E., and Aaronson, S. (2022). A very preliminary analysis of dall-e 2. *arXiv preprint arXiv:2204.13807*.

Martínez-Marín, T. and Duckett, T. (2005). Fast reinforcement learning for vision-guided mobile robots. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 4170–4175. IEEE.

Mazzi, G., Castellini, A., and Farinelli, A. (2021). Rule-based shielding for partially observable monte-carlo planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pages 243–251.

McCarty, L. T. (2002). Ownership: A case study in the representation of legal concepts. *Artificial Intelligence and Law*, 10(1-3):135–161.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Menache, I., Mannor, S., and Shimkin, N. (2002). Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European conference on machine learning*, pages 295–306. Springer.

Minsky, M. L. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34–34.

Moldovan, B., Moreno, P., and van Otterlo, M. (2013). On the use of probabilistic relational affordance models for sequential manipulation tasks in robotics. In *2013 IEEE International Conference on Robotics and Automation*, pages 1290–1295. IEEE.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al. (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597.

Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2007). Modeling affordances using bayesian networks. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4102–4107. IEEE.

Mordatch, I., Mishra, N., Eppner, C., and Abbeel, P. (2016). Combining model-based policy search with online model learning for control of physical humanoids. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 242–248. IEEE.

Morimoto, J. and Atkeson, C. (2002). Minimax differential dynamic programming: An application to robust biped walking. *Advances in neural information processing systems*, 15.

Muelling, K., Kober, J., and Peters, J. (2010). Learning table tennis with a mixture of motor primitives. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 411–416. IEEE.

Muggleton, S. (1991). Inductive logic programming. *New generation computing*, 8(4):295–318.

Mutz, F., Cardoso, V., Teixeira, T., Jesus, L. F., Golçalves, M. A., Guidolini, R., Oliveira, J., Badue, C., and De Souza, A. F. (2017). Following the leader using a tracking system based on pre-trained deep neural networks. In *2017 international joint conference on neural networks (IJCNN)*, pages 4332–4339. IEEE.

Mutz, F., Veronese, L. P., Oliveira-Santos, T., De Aguiar, E., Cheein, F. A. A., and De Souza, A. F. (2016). Large-scale mapping in complex field scenarios using an autonomous car. *Expert Systems with Applications*, 46:439–462.

Myers, A., Teo, C. L., Fermüller, C., and Aloimonos, Y. (2015). Affordance detection of tool parts from geometric features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1374–1381. IEEE.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018). Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31.

Najar, A. and Chetouani, M. (2021). Reinforcement learning with human advice: a survey. *Frontiers in Robotics and AI*, 8:584075.

Nguyen, D. Q., Nguyen, T. D., and Phung, D. (2022). Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 193–196, New York, NY, USA. Association for Computing Machinery.

Niekum, S., Chitta, S., Barto, A. G., Marthi, B., and Osentoski, S. (2013). Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9, pages 10–15607. Berlin, Germany.

Oberst, M. and Sontag, D. (2019). Counterfactual off-policy evaluation with gumbel-max structural causal models. *arXiv preprint arXiv:1905.05824*.

Pack, L. and Lozano-Pérez, T. (2011). Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1470–1477. IEEE.

Page, M. (2000). Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23(4):443–467.

Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., and Paschke, A. (2011). Legalruleml: Xml-based rules and norms. In *Rule-Based Modeling and Computing on the Semantic Web*, pages 298–312. Springer.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. (2013). Probabilistic movement primitives. *Advances in neural information processing systems*, 26.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE.

Patki, S., Daniele, A. F., Walter, M. R., and Howard, T. M. (2019). Inferring compact representations for efficient natural language understanding of robot instructions. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6926–6933. IEEE.

Payani, A. and Fekri, F. (2019). Inductive logic programming via differentiable deep neural logic networks. *arXiv preprint arXiv:1906.03523*.

Payani, A. and Fekri, F. (2020). Incorporating relational background knowledge into reinforcement learning via differentiable inductive logic programming. *arXiv preprint arXiv:2003.10386*.

Pearl, J. (2000). *Causality: models, reasoning and inference*, volume 29. Springer.

Pearl, J. (2018a). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

Pearl, J. (2018b). Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225. IEEE.

Petrovskaya, A., Perrollaz, M., Oliveira, L., Spinello, L., Triebel, R., Makris, A., Yoder, J.-D., Laugier, C., Nunes, U., and Bessière, P. (2012). Awareness of road scene participants for autonomous driving. *Handbook of Intelligent Vehicles*, pages 1383–1432.

Pfrimmer, D. (2009). teamwork and communication. *The Journal of Continuing Education in Nursing*, 40(7):294–295.

Plaat, A., Kosters, W., and Preuss, M. (2020). Deep model-based reinforcement learning for high-dimensional problems, a survey. *arXiv preprint arXiv:2008.05598*.

Prakash, B., Waytowich, N., Ganesan, A., Oates, T., and Mohsenin, T. (2020). Guiding safe reinforcement learning policies using structured language constraints. *UMBC Student Collection*.

Qin, M., Brawer, J., and Scassellati, B. (2021). Rapidly learning generalizable and robot-agnostic tool-use skills for a wide range of tasks. *Frontiers in Robotics and AI*, 8.

Qiu, J., Yang, Y., Wang, X., and Tao, D. (2021). Scene essence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8322–8333.

Ram, A. and Jones, E. K. (1994). *Foundations of foundations of artificial intelligence*. Department of Computer Science, Victoria University of Wellington.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G.,

Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. (2022). A generalist agent. *arXiv preprint arXiv:2205.06175*.

Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I. Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., et al. (2020). Logical neural networks. *arXiv preprint arXiv:2006.13155*.

Robinson, R. (1973). Counting labeled acyclic digraphs. In *New Directions in the Theory of Graphs: Proc of the Third Ann Arbor Conf. on Graph Theory*, pages 239–273. Academic Press.

Rocktäschel, T. and Riedel, S. (2017). End-to-end differentiable proving. *Advances in neural information processing systems*, 30.

Rodrigues, C., Gérard, P., Rouveirol, C., and Soldano, H. (2011). Active learning of relational action models. In *International Conference on Inductive Logic Programming*, pages 302–316. Springer.

Roncone, A., Mangin, O., and Scassellati, B. (2017). Transparent role assignment and task allocation in human robot collaboration. *Robotics and Automation (ICRA), IEEE International Conference on*.

Roth, A. M., Topin, N., Jamshidi, P., and Veloso, M. (2019). Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy. *arXiv preprint arXiv:1907.01180*.

Rozo, L., Jiménez, P., and Torras, C. (2013). Force-based robot learning of pouring skills using parametric hidden markov models. In *9th International Workshop on Robot Motion and Control*, pages 227–232. IEEE.

Rückert, U. (2008). *A statistical approach to rule learning*. PhD thesis, Technische Universität München.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. (2017). *Active preference-based learning of reward functions*.

SAE, I. (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE*.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*.

Sarathy, V. and Scheutz, M. (2016). A logic-based computational framework for inferring cognitive affordances. *IEEE Transactions on Cognitive and Developmental Systems*.

Sarathy, V., Scheutz, M., and Malle, B. F. (2017). Learning behavioral norms in uncertain and changing contexts. In *Cognitive Infocommunications (CogInfoCom), 2017 8th IEEE International Conference on*, pages 000301–000306. IEEE.

Sarathy, V., Wilson, J. R., Arnold, T., and Scheutz, M. (2016). Enabling basic normative hri in a cognitive robotic architecture. *arXiv preprint arXiv:1602.03814*.

Sarker, M. K., Zhou, L., Eberhart, A., and Hitzler, P. (2021). Neuro-symbolic artificial intelligence: Current trends. *arXiv preprint arXiv:2105.05330*.

Sattar, J. and Dudek, G. (2011). Towards quantitative modeling of task confirmations in human-robot dialog. In *2011 IEEE International Conference on Robotics and Automation*. IEEE.

Savarimuthu, B. T. R. and Cranefield, S. (2011). Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems. *Multiagent and Grid Systems*, 7(1):21–54.

Savarimuthu, B. T. R., Cranefield, S., Purvis, M. A., and Purvis, M. K. (2010). Obligation norm identification in agent societies. *Journal of Artificial Societies and Social Simulation*, 13(4):3.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Schaal, S. (2006). Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer.

Schaefer, A., Luft, L., and Burgard, W. (2018). Dct maps: Compact differentiable lidar maps based on the cosine transform. *IEEE Robotics and Automation Letters*, 3(2):1002–1009.

Scheutz, M., Krause, E., Oosterveld, B., Frasca, T., and Platt, R. (2017). Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1378–1386. International Foundation for Autonomous Agents and Multiagent Systems.

Scheutz, M., Schermerhorn, P., Kramer, J., and Anderson, D. (2007). First steps toward natural human-like hri. *Autonomous Robots*, 22(4):411–423.

Scheutz, M., Williams, T., Krause, E., Oosterveld, B., Sarathy, V., and Frasca, T. (2019). An overview of the distributed integrated cognition affect and reflection diarc architecture. *Cognitive architectures*, pages 165–193.

Scholz, J., Levihn, M., Isbell, C., and Wingate, D. (2014). A physics-based model prior for object-oriented mdps. In *International Conference on Machine Learning*, pages 1089–1097. PMLR.

Serafini, L. and Garcez, A. d. (2016). Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*.

Shah, D., Osinski, B., Ichter, B., and Levine, S. (2022). Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *arXiv preprint arXiv:2207.04429*.

Shah, J. and Breazeal, C. (2010). An empirical analysis of team coordination behaviors and action planning with application to human–robot teaming. *The Journal of the Human Factors and Ergonomics Society*, 52(2):234–245.

Shaker, M. R., Yue, S., and Duckett, T. (2009). Vision-based reinforcement learning using approximate policy iteration. In *2009 international conference on advanced robotics*, pages 1–6. IEEE.

Shi, S., Chen, H., Ma, W., Mao, J., Zhang, M., and Zhang, Y. (2020). Neural logic reasoning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1365–1374.

Siciliano, B. and Khatib, O. (2016). Robotics and the handbook. In *Springer Handbook of Robotics*, pages 1–6. Springer.

Silver, T., Athalye, A., Tenenbaum, J. B., Lozano-Perez, T., and Kaelbling, L. P. (2022). Learning neuro-symbolic skills for bilevel planning. *arXiv preprint arXiv:2206.10680*.

Şimşek, Ö. and Barto, A. G. (2004). Using relative novelty to identify useful temporal

abstractions in reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 95.

Sinapov, J. and Stoytchev, A. (2007). Learning and generalization of behavior-grounded tool affordances. In *2007 IEEE 6th International Conference on Development and Learning*, pages 19–24. IEEE.

Sinapov, J. and Stoytchev, A. (2008). Detecting the functional similarities between tools using a hierarchical representation of outcomes. In *2008 7th IEEE International Conference on Development and Learning*, pages 91–96. IEEE.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2022). Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

Smisek, J., Jancosek, M., and Pajdla, T. (2013). 3d with kinect. In *Consumer depth cameras for computer vision*, pages 3–25. Springer.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and brain sciences*, 11(1):1–23.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000a). *Causation, prediction, and search*. MIT press.

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000b). *Causation, prediction, and search*. MIT press.

Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K. E., Lian, Z., Gokmen, C., Buch, S., Liu, K., et al. (2022). Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pages 477–490. PMLR.

St Clair, A. and Mataric, M. (2015). How robot verbal feedback can improve team performance in human-robot task collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 213–220. ACM.

Stengel-Eskin, E., Hundt, A., He, Z., Murali, A., Gopalan, N., Gombolay, M., and Hager, G. (2022a). Guiding multi-step rearrangement tasks with natural language instructions. In *Conference on Robot Learning*, pages 1486–1501. PMLR.

Stengel-Eskin, E., Hundt, A., He, Z., Murali, A., Gopalan, N., Gombolay, M., and Hager, G. (2022b). Guiding multi-step rearrangement tasks with natural language instructions. In *Conference on Robot Learning*, pages 1486–1501. PMLR.

Stocking, K. C., Gopnik, A., and Tomlin, C. (2022). From robot learning to robot understanding: Leveraging causal graphical models for robotics. In *Conference on Robot Learning*, pages 1776–1781. PMLR.

Stolle, M. and Precup, D. (2002). Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer.

Storks, S., Gao, Q., Thattai, G., and Tur, G. (2021). Are we there yet? learning to localize in embodied instruction following. *arXiv preprint arXiv:2101.03431*.

Stoytchev, A. (2005). Behavior-grounded representation of tool affordances. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3071–3076. IEEE.

Stoytchev, A. (2008). Learning the affordances of tools using a behavior-grounded approach. In *Towards Affordance-Based Robot Control*, pages 140–158. Springer.

Stückler, J. and Behnke, S. (2014). Efficient deformable registration of multi-resolution surfel maps for object manipulation skill transfer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 994–1001. IEEE.

Sukhoy, V., Georgiev, V., Wegter, T., Sweidan, R., and Stoytchev, A. (2012). Learning to slide a magnetic card through a card reader. In *2012 IEEE International Conference on Robotics and Automation*, pages 2398–2404. IEEE.

Sun, R. (2004). Desiderata for cognitive architectures. *Philosophical psychology*, 17(3):341–373.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

Takano, W. and Nakamura, Y. (2012). Bigram-based natural language model and statistical motion symbol model for scalable language of humanoid robots. In *2012 IEEE International Conference on Robotics and Automation*, pages 1232–1237. IEEE.

Talamadupula, K., Briggs, G., Chakraborti, T., Scheutz, M., and Kambhampati, S. (2014). Coordination in human-robot teams using mental modeling and plan recognition. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2957–2962. IEEE.

Tan, Z. X., **Brawer**, t., and Scassellati, B. (2018). **that's mine! learning ownership relations and norms for robots** . *Thirty-second AAAI conference on artificial intelligence*.

Tang, C., Srishankar, N., Martin, S., and Tomizuka, M. (2021). Grounded relational inference: domain knowledge driven explainable autonomous driving. *arXiv preprint arXiv:2102.11905*.

Tellex, S., Knepper, R., Li, A., Rus, D., and Roy, N. (2014a). Asking for help using inverse semantics. In *Robotics: Science and Systems*.

Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*. AAAI Publications.

Tellex, S., Thaker, P., Joseph, J., and Roy, N. (2014b). Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285.

Thananjeyan, B., Balakrishna, A., Nair, S., Luo, M., Srinivasan, K., Hwang, M., Gonzalez, J. E., Ibarz, J., Finn, C., and Goldberg, K. (2021). Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922.

Thomason, J., Sinapov, J., Svetlik, M., Stone, P., and Mooney, R. J. (2016). Learning multi-modal grounded linguistic semantics by playing" i spy". In *IJCAI*, pages 3477–3483.

Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Toussaint, M. A., Allen, K. R., Smith, K. A., and Tenenbaum, J. B. (2018). Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science and Systems Foundation*.

Touzet, C. F. (1997). Neural reinforcement learning for behaviour synthesis. *Robotics and Autonomous Systems*, 22(3-4):251–281.

Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, F. P., Khemlani, S. S., and Schultz, A. C. (2013). Act-r/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 2(1):30–55.

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78.

Turner, J. (2019). Legal personality for AI. In *Robot Rules*, pages 173–205. Springer.

Ugur, E. and Piater, J. (2015). 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.

Ulbrich, S. and Maurer, M. (2013). Probabilistic online pomdp decision making for lane changes in fully automated driving. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2063–2067. IEEE.

Van Rooijen, J., Grondman, I., and Babuška, R. (2014). Learning rate free reinforcement learning for real-time motion control using a value-gradient based policy. *Mechatronics*, 24(8):966–974.

van Waveren, S., Pek, C., Tumova, J., and Leite, I. (2022). Correct me if i'm wrong: Using non-experts to repair reinforcement learning policies. In *Proceedings of the 17th ACM/IEEE International Conference on Human-Robot Interaction*, pages 1–9.

Vasconcelos, W. W., Kollingbaum, M. J., and Norman, T. J. (2009). Normative

conflict resolution in multi-agent systems. *Autonomous agents and multi-agent systems*, 19(2):124–152.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR.

Verma, P., Marpally, S. R., and Srivastava, S. (2021). Asking the right questions: Learning interpretable action models through query answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12024–12033.

Vogt, P. (2002). The physical symbol grounding problem. *Cognitive Systems Research*, 3(3):429–457.

Wachi, A. and Sui, Y. (2020a). Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, pages 9797–9806. PMLR.

Wachi, A. and Sui, Y. (2020b). Safe reinforcement learning in constrained markov decision processes. In *International Conference on Machine Learning*, pages 9797–9806. PMLR.

Wächter, M., Ovchinnikova, E., Wittenbeck, V., Kaiser, P., Szedmak, S., Mustafa, W., Kraft, D., Krüger, N., Piater, J., and Asfour, T. (2018). Integrating multi-purpose natural language understanding, robot's memory, and symbolic planning

for task execution in humanoid robots. *Robotics and Autonomous Systems*, 99:148–165.

Wahn, B., Schwandt, J., Krüger, M., Crafa, D., Nunnendorf, V., and König, P. (2016). Multisensory teamwork: using a tactile or an auditory display to exchange gaze information improves performance in joint visual search. *Ergonomics*, 59(6):781–795.

Wang, F.-Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., Zhang, J., and Yang, L. (2016a). Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120.

Wang, L., Jamieson, G. A., and Hollands, J. G. (2009). Trust and reliance on an automated combat identification system. *Human factors*, 51(3):281–291.

Wang, P. and Li, X. (2016). Different conceptions of learning: Function approximation vs. self-organization. In *International Conference on Artificial General Intelligence*, pages 140–149. Springer.

Wang, S. I., Liang, P., and Manning, C. D. (2016b). Learning Language Games through Interaction. In *Association for Computational Linguistics (ACL)*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Whitney, D., Eldon, M., Oberlin, J., and Tellex, S. (2016). Interpreting multimodal referring expressions in real time. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3331–3338. IEEE.

Wielemaker, J., Schrijvers, T., Triska, M., and Lager, T. (2012). Swi-prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96.

Wilde, N., Blidaru, A., Smith, S. L., and Kulić, D. (2020). Improving user specifications for robot behavior through active preference learning: Framework and evaluation. *The International Journal of Robotics Research*, 39(6):651–667.

Wilson, A., Fern, A., and Tadepalli, P. (2012). A bayesian approach for policy learning from trajectory preference queries. *Advances in neural information processing systems*, 25.

Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J., et al. (2017). A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46.

Wöhlke, J., Schmitt, F., and van Hoof, H. (2021). Hierarchies of planning and reinforcement learning for robot navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10682–10688. IEEE.

Wolcott, R. W. and Eustice, R. M. (2017). Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving. *The International Journal of Robotics Research*, 36(3):292–319.

Wong, J. M. and Grupen, R. A. (2016). Intrinsically motivated multimodal structure learning. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 260–261. IEEE.

Wookey, D. S. and Konidaris, G. D. (2015). Regularized feature selection in reinforcement learning. *Machine Learning*, 100(2):655–676.

Wray, R. E., Kirk, J. R., Laird, J. E., et al. (2021). Language models as a knowledge source for cognitive agents. *arXiv preprint arXiv:2109.08270*.

Wulfmeier, M., Abdolmaleki, A., Hafner, R., Springenberg, J. T., Neunert, M., Hertweck, T., Lampe, T., Siegel, N., Heess, N., and Riedmiller, M. (2019). Compositional transfer in hierarchical reinforcement learning. *arXiv preprint arXiv:1906.11228*.

Xie, A., Ebert, F., Levine, S., and Finn, C. (2019). Improvisation through physical understanding: Using novel objects as tools with visual foresight. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany.

Xiong, C., Shukla, N., Xiong, W., and Zhu, S.-C. (2016). 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.

Xu, Y., Liu, Z., Duan, G., Zhu, J., Bai, X., and Tan, J. (2022). Look before you leap: Safe model-based reinforcement learning with human intervention. In *Conference on Robot Learning*, pages 332–341. PMLR.

Xue, J.-r., Wang, D., Du, S.-y., Cui, D.-x., Huang, Y., and Zheng, N.-n. (2017). A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars. *Frontiers of Information Technology & Electronic Engineering*, 18(1):122–138.

Yang, J., Liu, Z., Xiao, S., Li, C., Lian, D., Agrawal, S., Singh, A., Sun, G., and Xie, X. (2021a). Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810.

Yang, X., Ji, Z., Wu, J., Lai, Y.-K., Wei, C., Liu, G., and Setchi, R. (2021b). Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Transactions on Neural Networks and Learning Systems*.

Yi, J. S. K., Kim, Y., and Chernova, S. (2022). Incremental object grounding using scene graphs. *arXiv preprint arXiv:2201.01901*.

Yu, Y., Chen, J., Gao, T., and Yu, M. (2019). Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR.

Zareian, A., Wang, Z., You, H., and Chang, S.-F. (2020). Learning visual commonsense for robust scene graph generation. In *European Conference on Computer Vision*, pages 642–657. Springer.

Zech, P., Haller, S., Lakani, S. R., Ridge, B., Ugur, E., and Piater, J. (2017). Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Zeylikman, S., Widder, S., Roncone, A., Mangin, O., and Scassellati, B. (2018). The hrc model set for human-robot collaboration research. In *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE.

Zhang, W.-N., Li, L., Cao, D., and Liu, T. (2018). Exploring implicit feedback for open domain conversation generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhao, L., Ichise, R., Yoshikawa, T., Naito, T., Kakinami, T., and Sasaki, Y. (2015). Ontology-based decision making on uncontrolled intersections and narrow roads. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 83–88. IEEE.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M.

(2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

Zhu, J. and Hastie, T. (2005). Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205.

Zhu, M., Liu, M., Shen, J., Zhang, Z., Chen, S., Zhang, W., Ye, D., Yu, Y., Fu, Q., and Yang, W. (2021). Mapgo: Model-assisted policy optimization for goal-oriented tasks. *arXiv preprint arXiv:2105.06350*.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE.

Zhuo, H. H., Yang, Q., Hu, D. H., and Li, L. (2010). Learning complex action models with quantifiers and logical implications. *Artificial Intelligence*, 174(18):1540–1569.

Ziegler, J., Werling, M., and Schroder, J. (2008). Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *2008 IEEE intelligent vehicles symposium*, pages 787–791. IEEE.