Interactive Policy Shaping for Human-Robot Collaboration with Transparent Matrix Overlays

Jake Brawer
Yale University
jake.brawer@yale.edu

Debasmita Ghose
Yale University
debasmita.ghose@yale.edu

Kate Candon
Yale University
kate.candon@yale.edu

Meiying Qin
Yale University
meiying.qin@yale.edu

Alessandro Roncone
University of Colorado Boulder
alessandro.roncone@colorado.edu

Marynel Vázquez
Yale University
marynel.vazquez@yale.edu

Brian Scassellati
Yale University
brian.scassellati@yale.edu

ABSTRACT

One important aspect of effective human–robot collaborations is the ability for robots to adapt quickly to the needs of humans. While techniques like deep reinforcement learning have demonstrated success as sophisticated tools for learning robot policies, the fluency of human-robot collaborations is often limited by these policies' inability to integrate changes to a user's preferences for the task. To address these shortcomings, 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 as an 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.

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**.

KEYWORDS

human-robot collaboration, symbolic reasoning, reinforcement learning, interactive robot learning

ACM Reference Format:
Jake Brawer, Debasmita Ghose, Kate Candon, Meiying Qin, Alessandro Roncone, Marynel Vázquez, and Brian 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 '23), March 13–16, 2023, Stockholm, Sweden.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3568162.3576983

HRI '23, March 13–16, 2023, Stockholm, Sweden

ACM ISBN 978-1-4503-9964-7/23/03.
https://doi.org/10.1145/3568162.3576983

1 INTRODUCTION

Human-robot collaboration (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 [27, 28, 36, 49, 51]. 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 [8, 19, 20, 35, 41], implicit [12, 30, 31, 53], multi-modal [9, 37, 45], and contextual [8, 24]. 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 [2], or subject to amendments during 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 [28, 33, 51]. 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 or learning strategies directly. The latter set of approaches has been referred to in recent years as action shielding [5, 15, 22, 23, 26, 32]. Action shielding is particularly relevant as it provides methods for restricting an agent's actions or providing particular action alternatives. These alternatives 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 execution-time HRC contexts. For example, one approach [47] defines three distinct

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

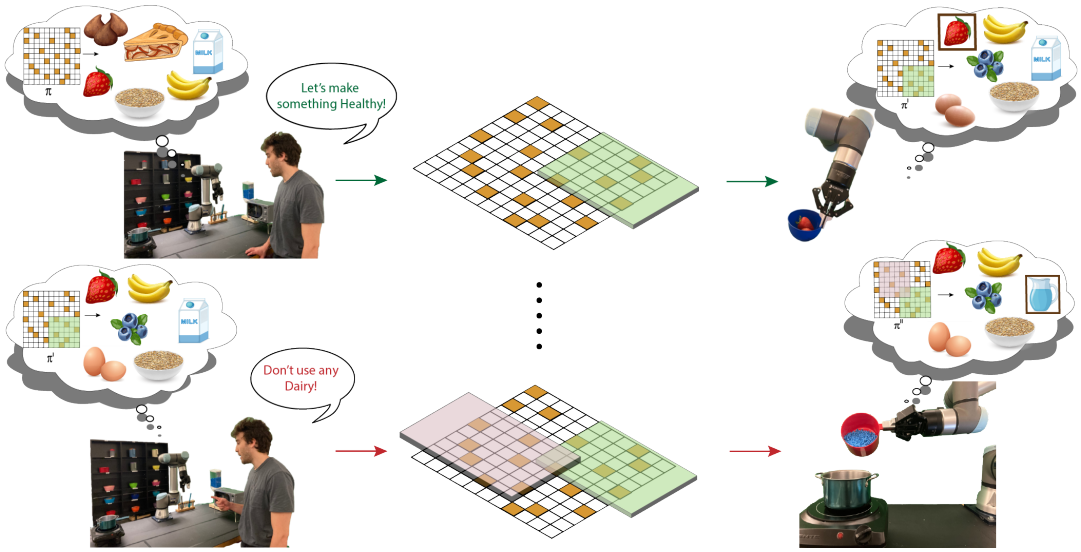

Figure 1: Transparent Matrix Overlays overview. Here the user prepares breakfast with a robot equipped with a base policy trained offline. The grids represent a Q-value matrix with high-value state-action pairs in orange. The user expresses their meal preference by providing a high-level directive *"let's 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.

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, requiring 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"*). Thus, we present a novel interactive policy-shaping approach we call the Transparent Matrix Overlay system that can extract and utilize symbolic rules from user-provided directives. Overlays act as mutable and composable high-level constraints on the robot's policy, allowing a user to quickly influence the policy in a naturalistic way without permanently altering it. The contributions of this paper are threefold:

(1) We describe a novel policy-shaping approach which we call Transparent Matrix Overlays (see Figure 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 [47] 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.

2 RELATED WORK

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

2.1 Robots Learning from Human Preferences

A growing area of research focuses on robots learning from human preferences [18, 24, 27, 28, 36, 49, 51]. Typical preference learning techniques query a human demonstrator to make a preference-based judgement between two candidate trajectories or actions [3, 4, 6, 27, 28, 36, 50], though they can also be inferred from user behavior [13]. From these selections, the system approximates a reward function using deep neural networks consistent with the expressed preferences [11]. Ibarz et al. [21] extended these common preference learning techniques by introducing additional forms of feedback such as user demonstrations. Recently, Lee et al. [27] 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 all 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. Therefore, 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. 5.1), they shape a policy at the level of the policy itself, meaning their effects are immediate and temporary. Thus, the proposed approach may be more practical in HRC contexts where such immediate behavioral adaptation is preferred.

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 [1, 14, 44, 52]. Shielded RL [5, 15, 22, 23, 26, 32, 47] 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 [14]. Closest to our work is the Shielded RL approach presented by [47], 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 [47] as our primary point of comparison in our evaluations.

2.3 Merging Symbolic Reasoning with RL

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 [34] and enabling neural networks to reason symbolically [38]. Typical approaches that merge symbolic reasoning with RL enable Deep RL-derived policies to be expressed using first-order logic [42], high-level programming languages [43], or by learning policies in an abstracted form of the underlying state space comprised of symbolic rules [7]. Symbolic state abstraction has also been employed to autonomously generate natural language explanations of learned robot policies [17]. 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 between 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.

3 PROPOSED APPROACH

Below we describe the Transparent Matrix Overlay approach (Fig. 1), which enables users to issue high-level directives to a robot resulting in modifications to its policy at execution time.

3.1 Preliminaries

We model a collaborative task with a modified goal-oriented MDP [29, 54] 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 $\pi_{\theta'}(a|s, \pi_\theta, D_{\theta'})$ 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.

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 $l_{pre} = 1$.

Algorithm 1: Modifying a policy by adding overlays

1 **Input:** policy π, state s, actions A, directive d, overlay list O
2 $\tau \leftarrow$ Confidence Threshold
3 l_{pre}, l_{post}, type $\leftarrow$ processDirective(d)
4 O.append((l_{pre}, l_{post}, type))
5 $O_{sorted} \leftarrow sortByType(O)$
6 $\hat{\pi} \leftarrow \pi$
7 **for** $a_i \in A$ **do**
8 **for** l_{pre}, l_{post}, *type* $\in O_{sorted}$ **do**
9 **if** *type == Prohibit* & $l_{pre} > \tau$ **then**
10 $\hat{\pi}(a_i|s) = (1 - l_{post}(a_i, s))\hat{\pi}(a_i|s)$
11 **else if** *type == Transfer* & $l_{pre} > \tau$ **then**
12 $l^*_{post} \leftarrow \max_{a_j \in A \setminus a_i} l_{post}(s, a_i, a_j)$
13 $\hat{\pi}(a_i|s) = (1 - l^*_{post})\hat{\pi}(a_i|s) + l^*_{post}\hat{\pi}(a_j|s)$
14 **else if** *type == Permit* & $l_{pre} > \tau$ **then**
15 $\hat{\pi}(a_i|s) = l_{post}(a_i, s)\hat{\pi}(a_i|s)$
16 **end**
17 **end**
18 $\hat{\pi}(A|s) \leftarrow \frac{\hat{\pi}(A|s)}{\sum_{a \in A} \hat{\pi}(a|s)}$
19 **Return:** $\hat{\pi}(A|s)$

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 1, 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 9 – 10 of Alg. 1) 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 $no\_completed\_dish(s) \implies \neg making\_pastry(a)$. Prohibitory overlays down-weight action probabilities resulting in actions that satisfy the rule being suppressed.

2. Transfer Overlays: Transfer overlays (lines 11 – 13 of Alg. 1) 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}. 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 14 – 15 of Alg. 1) implement rules that permit actions in states that satisfy the conditions of the overlay. These correspond to directives such as *"let's make cereal!"* and the rule $no\_completed\_dish(s) \implies making\_cereal(a)$. Permissive overlays up-weight satisfactory actions, making them more likely to be performed.

In this work, overlays are implemented as Prolog [48] 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 – 17 of Alg. 1). 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 5) 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 its 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.

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 [10, 40]. 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)?

4.1 Task and Problem Representation

Our task involves a user and a robot collaboratively preparing breakfast. 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 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 puffs" 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 complete when the side dish has been successfully 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) [46] consisting of an MLP with a single hidden layer, providing better initial 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.

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.

4.2.1 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) [16], 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 un-ordered (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. [47] 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 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 $no\_completed\_dish(s) \rightarrow making\_cereal(a)$,
- the side dish and its desired order, e.g., the permissive overlay $one\_completed\_dish(s) \rightarrow making\_pastry(a) \wedge sweet(a)$,
- the desired support type from the robot for a meal, e.g., the transfer overlay $\neg two\_completed\_meals(s) \rightarrow alternative(a1, a2) \wedge (making\_cereal(a1) \wedge do(a1)) \vee (making\_pastry(a1) \wedge say(a1))$.

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

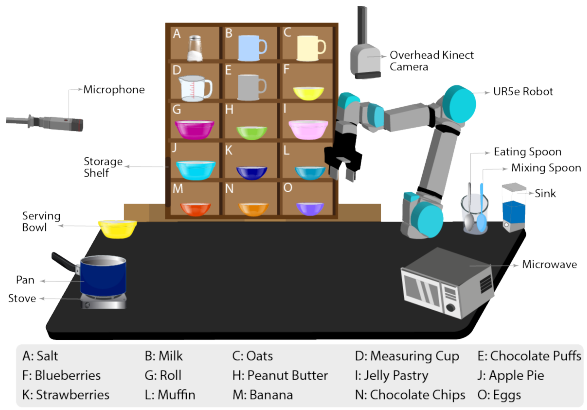

Figure 2: Setup for our Physical Robot Experiments

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.

4.2.2 Physical Robot. As shown in Figure 2, our physical setup consisted of a UR-5e robot, an overhead Microsoft Azure Kinect camera [39] 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 6, in the appendix for examples). As shown in the supplementary video, 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.

4.3 Reasoning and Learning

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

4.3.1 Rewards. During the physical and simulated 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. 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,
- $r = 1.0$, when the meal is fully complete

4.3.2 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.
- $C_{inter} = \{do(a), say(a), alternative(a1, a2)\}$ track whether an action is a do 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 [48].

5 RESULTS

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

5.1 Simulations

In our first set of experiments (Fig. 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 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 (Fig. 4 and Tables 3-5 in the appendix), 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 Fig. 3), the shield model performs the most

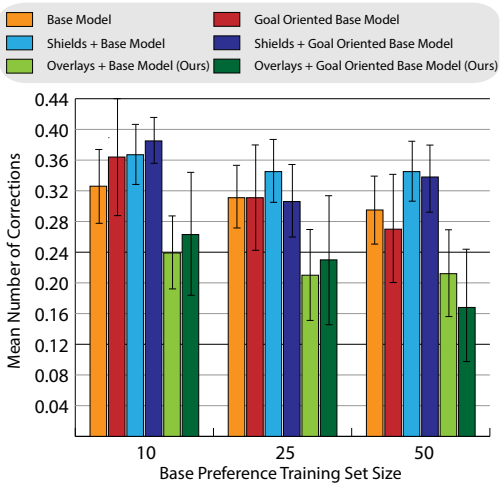

Figure 3: Comparing adaptation to new meal preferences based on initial preference training set size.

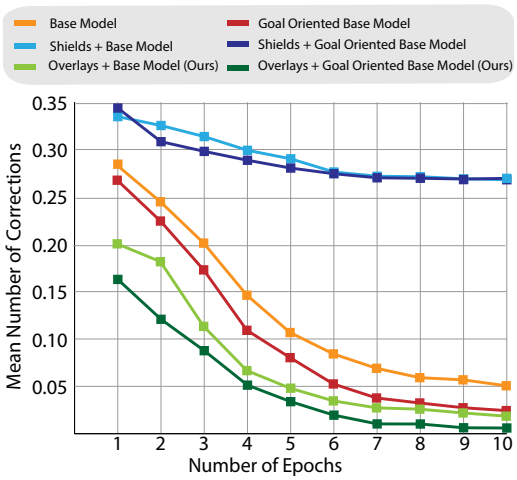

Figure 4: Comparing new meal preference learning.

poorly and notably does not improve substantially despite repeated exposure to the same meal preferences across the 10 episodes.

5.2 Physical Robot

We show three proof-of-concept experiments using a real robot and 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. 4.1), modeling user uncertainty regarding the course of the collaboration. Thus, in these physical robot experiments summarized in Fig. 5 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 (Fig. 5 b). For all experiments, we used the same goal-conditioned base policy trained on 50 randomly sampled synthetic meal collaborations (see sec.4.2.1) for 500 episodes, the observed convergence point of the learning.

Fig. 5 (a) 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-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.

Fig. 5 (b) summarizes the second experiment. The user initially prompts the robot with the directive *"first let's make oatmeal, then a pastry,"* communicating both a 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 a two mistakes, while the base model makes 6 incorrect predictions.

Fig. 5 (c) 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.

6 DISCUSSION

In this paper, 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 perferences?*
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

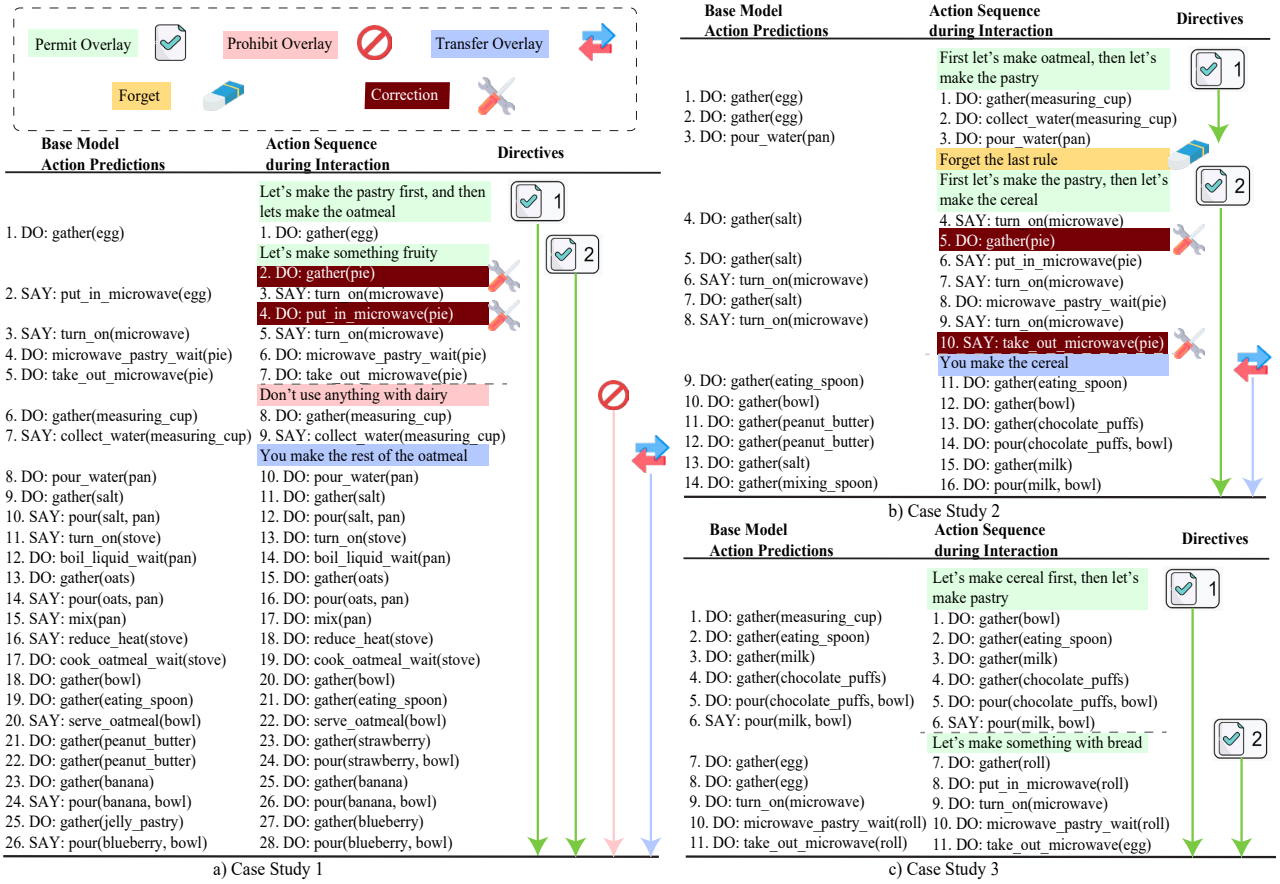

Figure 5: Results of the Case Study with the Physical Robot Utilizing Overlays: The first column of case studies, a), b), and c) depict the action sequence predicted by the base model. The second column shows the action sequence of the policy modified by the overlays and corrective actions, and the third column graphically depicts the applied overlay's activation intervals.

were applied to the goal-oriented base model compared to all other models (refer to Fig. 4), suggesting that overlay-equipped models have the potential to most effectively leverage this information.
(4) *How do overlays compare to other methods regarding (1-3) ?* Surprisingly, our adapted version of the shielded approach from [47] consistently under-performed w.r.t. the baseline policy despite explicitly encoding information about the updated preferences. Figure 5 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 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 do limit the flexibility of our approach. Future work should incorporate autonomous methods for state and action abstraction using methods such as those described in Konidaris et al. [25] and Ugur et al. [42]. 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.

ACKNOWLEDGMENTS

This work was partially supported by the Office of Naval Research (ONR) award No. N00014-18-1-2776 and National Science Foundation (NSF) under grants No. 2033413, 1955653, 1928448, 1936970, 1813651, 2106690, and IIS-2106690. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or ONR.

REFERENCES

[1] Joshua Achiam et al. "Constrained policy optimization". In: *International conference on machine learning*. PMLR. 2017, pp. 22–31.

[2] Pulkit Agrawal. "The Task Specification Problem". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1745–1751.

[3] Riad Akrour, Marc Schoenauer, and Michèle Sebag. "April: Active preference learning-based reinforcement learning". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2012, pp. 116–131.

[4] Riad Akrour, Marc Schoenauer, and Michele Sebag. "Preference-based policy learning". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2011, pp. 12–27.

[5] Mohammed Alshiekh et al. "Safe reinforcement learning via shielding". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[6] Erdem Biyik and Dorsa Sadigh. "Batch active preference-based learning of reward functions". In: *Conference on robot learning*. PMLR. 2018, pp. 519–528.

[7] Nicolas Bougie and Ryutaro Ichise. "Towards Interpretable Reinforcement Learning with State Abstraction Driven by External Knowledge". In: *IEICE Transactions on Information and Systems* 103.10 (2020), pp. 2143–2153.

[8] Jake Brawer et al. "Situated human–robot collaboration: predicting intent from grounded natural language". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 827–833.

[9] Arthur Bucker et al. "Reshaping Robot Trajectories Using Natural Language Commands: A Study of Multi-Modal Data Alignment Using Transformers". In: *arXiv preprint arXiv:2203.13411* (2022).

[10] Micah Carroll et al. "On the utility of learning about humans for human-ai coordination". In: *Advances in neural information processing systems* 32 (2019).

[11] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Advances in neural information processing systems* 30 (2017).

[12] Yuchen Cui et al. "The empathic framework for task learning from implicit human feedback". In: *arXiv preprint arXiv:2009.13649* (2020).

[13] Tyler Frasca et al. "Enabling fast instruction-based modification of learned robot skills". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 7. 2021, pp. 6075–6083.

[14] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[15] Mirco Giacobbe et al. "Shielding Atari games with bounded prescience". In: *arXiv preprint arXiv:2101.08153* (2021).

[16] Bradley Hayes and Brian Scassellati. "Autonomously constructing hierarchical task networks for planning and human-robot collaboration". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 5469–5476.

[17] Bradley Hayes and Julie A Shah. "Improving robot controller transparency through autonomous policy explanation". In: *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*. IEEE. 2017, pp. 303–312.

[18] Joey Hejna and Dorsa Sadigh. "Few-Shot Preference Learning for Human-in-the-Loop RL". In: *arXiv preprint arXiv:2212.03363* (2022).

[19] Wenlong Huang et al. "Inner Monologue: Embodied Reasoning through Planning with Language Models". In: *arXiv preprint arXiv:2207.05608* (2022).

[20] Wenlong Huang et al. "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents". In: *arXiv preprint arXiv:2201.07207* (2022).

[21] Borja Ibarz et al. "Reward learning from human preferences and demonstrations in atari". In: *Advances in neural information processing systems* 31 (2018).

[22] Nils Jansen et al. "Safe reinforcement learning via probabilistic shields". In: *arXiv preprint arXiv:1807.06096* (2018).

[23] Nils Jansen et al. "Shielded decision-making in MDPs". In: *arXiv preprint arXiv:1807.06096* (2018).

[24] Yash Kant et al. "Housekeep: Tidying Virtual Households using Commonsense Reasoning". In: *arXiv preprint arXiv:2205.10712* (2022).

[25] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. "From skills to symbols: Learning symbolic representations for abstract high-level planning". In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 215–289.

[26] Bettina Könighofer et al. "Online shielding for stochastic systems". In: *NASA Formal Methods Symposium*. Springer. 2021, pp. 231–248.

[27] Kimin Lee, Laura Smith, and Pieter Abbeel. "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training". In: *arXiv preprint arXiv:2106.05091* (2021).

[28] Kimin Lee et al. "B-pref: Benchmarking preference-based reinforcement learning". In: *arXiv preprint arXiv:2111.03026* (2021).

[29] Minghuan Liu, Menghui Zhu, and Weinan Zhang. "Goal-conditioned reinforcement learning: Problems and solutions". In: *arXiv preprint arXiv:2201.08299* (2022).

[30] Robert Loftin et al. "Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning". In: *Autonomous agents and multi-agent systems* 30.1 (2016), pp. 30–59.

[31] Olivier Mangin, Alessandro Roncone, and Brian Scassellati. "How to be Helpful? Supportive Behaviors and Personalization for Human-Robot Collaboration". In: *Frontiers in Robotics and AI* (2022), p. 426.

[32] Giulio Mazzi, Alberto Castellini, and Alessandro Farinelli. "Rule-based Shielding for Partially Observable Monte-Carlo Planning". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 31. 2021, pp. 243–251.

[33] Anis Najar and Mohamed Chetouani. "Reinforcement learning with human advice: a survey". In: *Frontiers in Robotics and AI* 8 (2021), p. 584075.

[34] Ali Payani and Faramarz Fekri. "Incorporating relational background knowledge into reinforcement learning via differentiable inductive logic programming". In: *arXiv preprint arXiv:2003.10386* (2020).

[35] Bharat Prakash et al. "Guiding safe reinforcement learning policies using structured language constraints". In: *UMBC Student Collection* (2020).

[36] Dorsa Sadigh et al. *Active preference-based learning of reward functions*. 2017.

[37] Dhruv Shah et al. "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action". In: *arXiv preprint arXiv:2207.04429* (2022).

[38] Shaoyun Shi et al. "Neural logic reasoning". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 1365–1374.

[39] Jan Smisek, Michal Jancosek, and Tomas Pajdla. "3D with Kinect". In: *Consumer depth cameras for computer vision*. Springer, 2013, pp. 3–25.

[40] Sanjana Srivastava et al. "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments". In: *Conference on Robot Learning*. PMLR. 2022, pp. 477–490.

[41] Elias Stengel-Eskin et al. "Guiding Multi-Step Rearrangement Tasks with Natural Language Instructions". In: *Conference on Robot Learning*. PMLR. 2022, pp. 1486–1501.

[42] Emre Ugur and Justus Piater. "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2627–2633.

[43] Abhinav Verma et al. "Programmatically interpretable reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 5045–5054.

[44] Akifumi Wachi and Yanan Sui. "Safe reinforcement learning in constrained Markov decision processes". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9797–9806.

[45] Basil Wahn et al. "Multisensory teamwork: using a tactile or an auditory display to exchange gaze information improves performance in joint visual search". In: *Ergonomics* 59.6 (2016), pp. 781–795. URL: http://dx.doi.org/10.1080/00140139.2015.1099742.

[46] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[47] Sanne van Waveren et al. "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*. 2022, pp. 1–9.

[48] Jan Wielemaker et al. "Swi-prolog". In: *Theory and Practice of Logic Programming* 12.1-2 (2012), pp. 67–96.

[49] Nils Wilde et al. "Improving user specifications for robot behavior through active preference learning: Framework and evaluation". In: *The International Journal of Robotics Research* 39.6 (2020), pp. 651–667.

[50] Aaron Wilson, Alan Fern, and Prasad Tadepalli. "A bayesian approach for policy learning from trajectory preference queries". In: *Advances in neural information processing systems* 25 (2012).

[51] Christian Wirth et al. "A survey of preference-based reinforcement learning methods". In: *Journal of Machine Learning Research* 18.136 (2017), pp. 1–46.

[52] Yunkun Xu et al. "Look Before You Leap: Safe Model-Based Reinforcement Learning with Human Intervention". In: *Conference on Robot Learning*. PMLR. 2022, pp. 332–341.

[53] Wei-Nan Zhang et al. "Exploring implicit feedback for open domain conversation generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[54] Menghui Zhu et al. "Mapgo: Model-assisted policy optimization for goal-oriented tasks". In: *arXiv preprint arXiv:2105.06350* (2021).