Robot Behavior Personalization from Sparse User Feedback

Maithili Patel[1], Sonia Chernova[1]

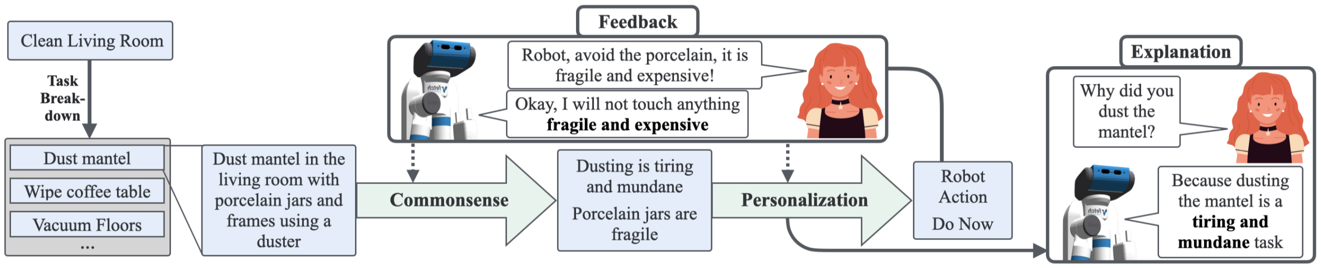

Clean Living Room
Task Break-down
Dust mantel
Wipe coffee table
Vacuum Floors
...
Dust mantel in the living room with porcelain jars and frames using a duster
Commonsense
Feedback
Robot, avoid the porcelain, it is fragile and expensive!
Okay, I will not touch anything **fragile and expensive**
Dusting is tiring and mundane
Porcelain jars are fragile
Personalization
Robot Action
Do Now
Explanation
Why did you dust the mantel?
Because dusting the mantel is a **tiring and mundane** task

Fig. 1. Task Adaptation using Abstract Concepts (TAACo) learns user preferences regarding how they want the robot to assist with an open set of household tasks from limited user feedback. This requires commonsense reasoning to extract relevant semantic information regarding a novel task, and personalization based on limited feedback. In addition, TAACo can explain its predictions to the user in an intuitive manner.

***Abstract*—As service robots become more general-purpose, they will need to adapt to their users' preferences over a large set of *all* possible tasks that they can perform. This includes preferences regarding which actions the users prefer to delegate to robots as opposed to doing themselves. Existing personalization approaches require task-specific data for each user. To handle diversity across all household tasks and users, and nuances in user preferences across tasks, we propose to learn a task adaptation function independently, which can be used in tandem with any universal robot policy to personalize robot behavior. We create Task Adaptation using Abstract Concepts (TAACo) framework. TAACo can learn to predict the user's preferred manner of assistance with any given task, by mediating reasoning through a representation composed of abstract concepts built based on user feedback. TAACo can generalize to an open set of household tasks from small amount of user feedback and explain its inferences through intuitive concepts. We evaluate our model on a dataset we collected of 5 people's preferences, and show that TAACo outperforms GPT-4 by 16% and a rule-based system by 54%, on prediction accuracy, with 40 samples of user feedback**

***Index Terms*—Human-Centered Robotics, Domestic Robotics**

I. Introduction

SERVICE robots have the potential to assist diverse users across warehouses, homes, assisted living facilities, etc., by performing tasks based on user commands [1]–[4]. However, these methods require unambiguous commands [1], [2], such as 'pick up the cereal box from the pantry and bring it here', or need the user to resolve ambiguities during operation [3], [4]. For complex, high-level tasks, such as 'help me with breakfast', unambiguous definition of every detail is impractical, and robots must autonomously adapt to user preferences to assist with under-specified tasks. Such adaptation includes determining 'when' a task should be done, 'how' it should be done, and 'who' should do it. Prior works have explored questions of anticipating 'when' a task should be done [5]–[7], and details of 'how' the user prefers a task to be executed [8]–[11], but the final question of 'who' should do a given task, between humans and robots [12]–[15], is relatively under-explored in the context of personal household preferences. This problem involves two main challenges: the diversity in preferences across users, and diversity across the vast space of tasks that a robot might encounter. Given the growing demand for robotic support in homes and assisted living facilities, we focus on older adults, a demographic that prior studies [16] have shown differ in their preferences regarding high-level activities, such as *meal preparation*, that they prefer delegating to a robot. We examine preferences over more granular tasks, such as *wiping the table* and *preparing coffee*, through a case study involving interviews with five older adults, and find low agreement on preferences between users, indicated by an average pairwise Cohen's Kappa of 0.23, underscoring the need for personalization to each user.

Although users prefer to customize robots [17], existing methods on task allocation [12], [13] in human-robot teaming primarily focus on optimizing task efficiency rather than user preferences, and methods that consider user preferences [14], [15] require exhaustive annotation or prior experience across the space of tasks, which becomes intractable for general robots that can perform a multitude of tasks. If users were required to set explicit rules for the preferences we gathered, they would require on average 76.5 rules for every 100 scenarios. Our **key insight** is that while the space of all actions is diverse and intractable, people's preferences associated with them are anchored in a relatively smaller space of abstract concepts (e.g. fragility of an object, mundaneness of a task, etc.). Moreover, an understanding of such abstract concepts can be bootstrapped from prior knowledge sources, and utilizing them in an intermediate representation helps the robot align its reasoning with the user's, and leverage user-generated explanations to further improve such alignment. Ultimately, this not only helps generalization, but enables such a model to intuitively explain model inferences to the user.

We contribute TAACo, which leverages prior knowledge

This paper was recommended for publication by Editor Gentiane Venture upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by NSF IIS 2112633 and Amazon Research.

[1]Both authors are with School of Interactive Computing, Georgia Institute of Technology, USA maithili@gatech.edu

to create a semantically-rich low-dimensional intermediate representation to personalize robot behavior to align with user preferences from sparse user feedback. We define robot behavior personalization as adaptation of any given task, by determining who should perform it, and any communication involved in this allocation. In this regard, TAACo can adapt *any* robot task, generated by *any* universal policy, to the particular user, from small amounts of verbal user feedback on prior tasks. For instance, a user's preference that the robot should prepare ingredients for an omelet but not cook it, can be inferred from past feedback, such as asking it to not cook pasta, and requesting help with tasks like clearing the kitchen counter after cooking. For a given task, TAACo can help a robot decide whether it should perform the action, and whether it should do so right now or later, or should it allow the user to execute it, and whether it should remind them about it. It also accounts for known constraints on capabilities of the user or robot, which might prevent them from being able to execute a task, in addition to user preferences in making this decision. Moreover, TAACo trains only on target user's data, eliminating the need for data sharing, and allowing data to be stored and used locally.

The TAACo framework[1], outlined in Figure 1, can learn to predict the preferred robot behavior adaptation for any given task, by reasoning through a space of abstract concepts. Concretely, it offers the following key advancements

1) The ability to generalize user preferences to an open set of tasks from a small amount of user feedback
2) The ability to explain its decisions faithfully through abstract concepts that a user can understand

Evaluations on a custom dataset[2] show that TAACo outperforms GPT-4 by 16% and a rule-based system by 54%, on accuracy of correctly predicting the preferred manner of assistance based on 40 samples of user feedback.

II. Background and Related Work

In this section, we discuss prior work on personalizing robot behavior to user preferences and bootstrapping prior knowledge contained in Large Language Models (LLMs).

A common approach to personalization of robot tasks is for users to directly codify their preferences through end-user programming methods. Interfaces that support 'if...then...' style rules based on abstract semantic conditions, such as *food is being prepared* [17], or IoT sensor triggers [18], have enabled personalized scheduling of behaviors. However, giving users total control over robot actions, requires them to be accurate and exhaustive in defining their expectations. For complex behaviors, user-specification of tasks have been shown to fail due to unforeseen repercussions [19], [20].

More generally, personalization in human robot interaction involves adapting to user's task preferences on 'how' and 'when' a task should be executed, as well as 'who' should execute a given task. Towards addressing the 'how' question, prior works have learned to adapt to the user's preferred manner of executing various tasks, such as learning preferred trajectories, object locations, or action ordering, based on demonstrations [8]–[10], interactive feedback [11], [21], [22], facial expressions [23], or scene context [24], [25]. Prior works have also explored the 'when' question to enable proactivity through behavior prediction. In collaborative settings [7], [26]–[28], such as robot handovers, short-term predictions have been used to improve task efficiency and fluency, and in household assistance [5], [6] long-term predictions have enabled timely robot assistance without direction.

The final question of accounting for user preferences while determining 'who' a task should be allocated to, remains relatively under-explored in the context of household robots. The field of task allocation has extensively addressed assignment of tasks in multi-robot and human-robot teams [12]–[14]. However, existing works primarily weigh the utilities and costs of various allocations, aiming to optimize the efficiency of achieving task goals. While this is useful in factory settings, in domestic settings, the goal shifts from task efficiency to promoting user comfort and satisfaction. In such settings, the robot must understand allocation preferences of which tasks the users *prefer* doing themselves, as opposed to delegating to a robot. Closest to our work are methods that seek to recover hidden human preferences, by learning a reward function [15], [29]. But these works are limited to pre-specified tasks and require past collaboration experience with the particular user on those tasks. Transferring user preferences across tasks has been investigated towards optimizing handovers [28], and expanding user capabilities [30]. We aim to generalize allocation preferences of the user across an open set of tasks.

Finally, foundational models have benefited various robotic applications, but most related to our work is task-related assistance. By enabling direct user-robot interaction, these models have supported teaching various tasks to the robot [4], natural instruction following [31], [32], queryable scene representations [33], [34], and explaining robot failures [35]. Knowledge of human norms encoded in LLMs has been used to personalize robot behavior [36] and to model humans [37]. These works employ the LLMs to directly perform the entire target task [35], [37] or to solve parts of it [4], [36]. Alternatively, prior works [2], [38] have used LLMs to embed inputs into semantically rich latent spaces. In addition to using latent embeddings, we use an LLM to create an intermediate representation composed of explicit abstract concepts.

III. Problem Formulation

The primary aim of this work is to learn a model Φ to predict the desired task adaptation ϕ to regulate execution of $\boldsymbol{t}$ to match user preferences and user's or robot's capability constraints, and produce an explanation $\mathcal{E}_{robot}$ for its prediction. We assume that the high-level robot behavior of the robot is governed by a plan or policy π, which suggests task $\boldsymbol{t}$ that the robot should perform, given world state $\boldsymbol{s}$, and robot's goal. As the robot performs various tasks $\boldsymbol{t}$ in the household, produced by π either proactively or reaction to a command, the user can provide feedback u, about their preferred adaptations. Over time, using u, the robot must learn a function Φ to predict the preferred task adaptation ϕ for novel tasks that it encounters.

[1]The code for TAACo is available at https://github.com/Maithili/TAACO
[2]The dataset is available at https://github.com/Maithili/TAACO_data.git

TABLE I. Data collected and parsed from user interviews.

	User Utterance	Task Description $\boldsymbol{t} = (c^a, c^h, \{c^o\}, \{c^l\})$	State Constraint S_c	Task Adaptation ϕ
1	*"Yep, same [as other cooking tasks], it should do it. I don't like cooking."*	Pouring oil in a pan to cook food, Preparing a Meal, {Oil Bottle, Pan, Stove}, {Kitchen}	-	*do_now*
2	*"No, the robot shouldn't ever operate the stove; it is a serious fire hazard."*	Turning on the stove, Preparing a Meal, {Stove, Stove knobs}, {Kitchen}	-	*no_action*
3	*"Yes, I would love for the robot to do such mundane activities... But it shouldn't cause ruckus when I'm asleep."*	Arranging pots and pans in the kitchen shelves, Organizing the Kitchen, {Pots, Pans, Kitchen Shelves}, {Kitchen}	user not asleep	*do_now*
			user is asleep	*do_later*
4	*"Sure, the robot can do it... I trust the robot to follow a recipe when baking."*	mixing cake batter to bake a birthday cake, baking, {cake batter, mixing bowl, wooden spoon}, {kitchen}	-	do_now
5	*"No, I'm particular about cooking and such, I'd rather do it myself."*	mixing cake batter to bake a birthday cake, baking, {cake batter, mixing bowl, wooden spoon}, {kitchen}	-	no_action
6	*"I don't mind the robot folding my laundry... But it is tight around my closet, so if I'm there, I wouldn't like it moving around me so much."*	Folding and putting away clothes in the dresser, Doing Laundry, {Pants, Shirt, Jackets, Dresses, Dresser}, {Closet}	user is nearby	*do_later*
			user not nearby	*do_now*

We represent task $\boldsymbol{t}$, through a tuple of the components describing it, $\boldsymbol{t} = (c^a, c^h, \{c^o\}, \{c^l\})$. c^a is an action description, e.g. *dusting the mantel*. c^h is the corresponding high-level activity, e.g. *cleaning the living room*. $\{c^o\}$ is a set of objects involved in the action, e.g. {*mantel, duster, porcelain jar, photo frame*}. $\{c^l\}$ is a set of locations involved in executing the action, e.g. {*living room*}. We allow each component to be any natural language phrase and so the task is not confined to a closed set, and can be generated by open-set language-based planners, as well as classical or RL-based planners for which language labels are available. The state $\boldsymbol{s} \in S$ is comprised of a set of binary variables $\boldsymbol{s} = \{s_i\}$, and can include predicates such as 'user is asleep', 'guests are present', 'weekend' etc.

We use task adaptation $\Phi(\boldsymbol{t}, \boldsymbol{s}) \rightarrow \phi$ to represent adaptation of robot behavior over a task $\boldsymbol{t}$. Task adaptation ϕ can be one of acting on task $\boldsymbol{t}$ immediately (do_now), acting on $\boldsymbol{t}$ later (do_later), reminding the user to do $\boldsymbol{t}$ (remind), and not doing anything about $\boldsymbol{t}$ (no_action). This set of four adaptations is based on preferences people expressed in our pilot case study (Section V-A), but it can be extended as needed. A function Φ which predicts a task adaptation is used to personalize robot behavior in one of two ways: as a filter to determine whether or not to execute the action selected by π, or as an external constraint to π, allowing it to optimize for user preferences, similar to user schedules [39] or motion constraints [40].

To adapt a given task, the robot has access to user feedback u over prior tasks (examples shown in Table I), and optionally, some context about constraints on user's capability in natural language (e.g. has asthma and is sensitive to dust), and which tasks π can support. User feedback samples $(\phi, \boldsymbol{t}, S_c, \mathcal{E}_{user})$ include the preferred task adapter ϕ for a task $\boldsymbol{t}$, and can optionally include a state constraint S_c and user-generated explanation $\mathcal{E}_{user}$. State constraint S_c identifies a subset of the state space where the preference applies. The user-generated explanation, (examples shown in Table II), expresses the user's reasoning behind the given preference $\mathcal{E}_{user} = \{(c^x, \theta^x)\}$, and includes a set of abstract concepts θ^x associated with components c^x of the task, for any type of component $x \in \{a, h, o, l\}$. Similar to the user generated explanation, the robot should also be able to offer an explanation $\mathcal{E}_{robot} = \Psi(\Phi, \boldsymbol{t}, \boldsymbol{s})$ for its prediction.

IV. Method

The Task Adaptation using Abstract Concepts framework (TAACo), addresses the two main challenges in learning Φ, portrayed in Figure 1: 1) generalization to an open-set of tasks by extracting relevant semantic information, and 2) personalized prediction of preferred task adaptation based on limited feedback. As outlined in Figure 2, TAACo is composed of a Commonsense module and a Personalization module to address the two challenges. Based on the idea that people's preferences are grounded in abstract semantic concepts, such as *fragility* of objects or *mundaneness* of tasks, we use such concepts to create an intermediate representation between the two modules. This representation helps explain model decisions in an intuitive manner, and leverage user-generated explanations to align the personalization model with the user. Unlike prior works [41], [42], we do not restrict the set of concepts to a predefined closed set, instead allowing the user to define new concepts as needed through natural language.

The aim of the **Commonsense Module** is to extract relevant semantic information about the task $\boldsymbol{t}$ into an intermediate representation $\tilde{t} = \{(x, \theta^x, m)\}$. To create $\tilde{t}$ we distill semantic information from each task component c^x, which can be of one of four component types $x \in \{a, h, o, l\}$ (action, high level activity, object, location), by predicting a score m of how well it matches each of a set of relevant abstract concepts $\{\theta^x\}$. For every θ^x-c^x (concept-component) pair, we prompt GPT-4 to predict a score on a scale of 1-10, and linearly rescale it to [0,1], to obtain m. We use this to compose a tuple (x, θ^x, m), such as $(\text{object}, \text{is\_fragile}, 0.85)$ for $c_x = \text{mug}$ and $\theta^x = \text{is\_fragile}$. Each θ^x-c^x match prompt is independent of other task components, concepts or prior feedback, allowing responses to be cached and reused across tasks. We create an initial set of concepts for each component type, emulating a factory configuration, and adapt to each user by adding new concepts obtained from user feedback to these sets.

The **Personalization Module** predicts the preferred adaptation for each task, given the task and world state. The task input is represented in the form of abstract concepts $\tilde{t} = \{(x, \theta^x, m)\}$, and the world state as a set of binary variables $S = \{s_i\}$. We first embed each input element of task representation and state variables, individually into a set of three latent vectors: a type embedding h_x, a concept embedding h_θ, and a score embedding h_m. The three vectors are concatenated to form inputs to a transformer encoder with a classification head to predict the final task adaptation.

For the task representation, we embed each element of the tuple (x, θ^x, m) to create h_x, h_θ and h_m. We create h_x by

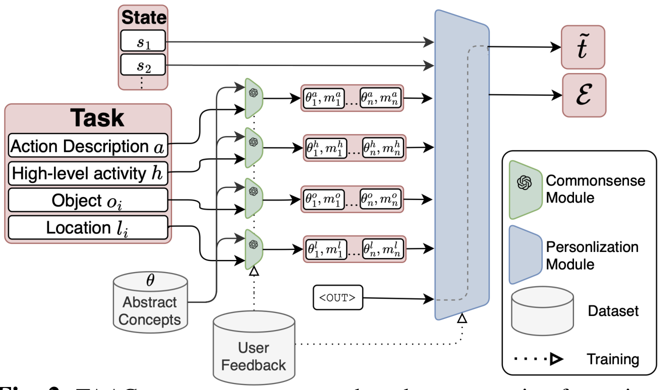


Fig. 2. TAACo generates a concept-based representation for a given task through a Commonsense Module, and uses it to predict the preferred action adaptation through a Personalization Module.

learning custom embedding vectors to identify each component type $x \in \{a, h, o, l\}$. We create $h_\theta = \rho_c(\gamma_{LLM}(\theta))$ using an off-the-shelf language embedder γ_{LLM}, and projecting the resulting embedding into latent space using a learned concept projection function ρ_c. We create $h_m = \rho_m(m)$ using a feedforward magnitude projection function ρ_m to project the match scores m. To embed the state variables, we learn custom embeddings h_θ for each binary state variable $s_i \in S$, and a single embedding for h_x. To create h_m, we reuse ρ_m to project the binary state value $h_m = \rho_m(m),\ m \in \{0, 1\}$. We process the final latent embedding for each input component $h_x|h_\theta|h_m$, along with an output token, `<OUT>`, through a transformer encoder. We use the output feature at `<OUT>` to classify the desired task adaptation through a classification head. We train the Personalization Module in a supervised manner using preference data obtained from the user. We train the model for the four adaptations in our data, but it can be extended to accommodate additional adaptations by modifying the dimensions of the classifier head. Note that the user is not required to use every available adaptation.

We **generate model explanations** using the human-interpretable intermediate representation, and **align the model's reasoning with the user's** by explicitly training the model's explanations. Such training not only improves the model's ability to explain, but also improves its ability to generalize to new actions, by learning the correct underlying structure of the data and avoiding spurious correlations. Results in Section VI-D empirically verify this claim.

We define the space of all explanations to be the set of all inputs to the Personalization Module, namely all component-concept pairs that form the task representation and all state variables $\mathcal{E} \in \{(c^x, \theta^x)\} \cup \{s\}$. Extracting an explanation is equivalent to finding the most important component. We use attention weights w_a between the output token `<OUT>` and each input to compute the probability of that input component to be the explanation as $1 - e^{-w_a}$. The input component with the highest probability is used as the explanation. We add an auxiliary loss to motivate the explanation probabilities to align with user feedback, through a binary cross entropy loss evaluated based on whether each input component is a part of explanations $\mathcal{E}_{user}$ or constraints $s \in S_c$ provided by the user for that task. We use attention weights w_a from the final layer of the transformer encoder to allow context from other inputs to influence the attention weights through previous layers.

Finally, capability constraints of the user and the robot might render some adaptations unusable. If information about these capabilities is available, we infer whether the user or robot can perform the task and constrain the set of valid adaptations accordingly. We assume that the robot policy π applies to only those tasks which it can perform, allowing it to indicate task support. To assess user's ability to do a given task, we prompt GPT-4 for a binary yes/no response, using known context about the user. If the robot cannot perform the task, we constrain the valid options to 'no_action' and 'remind'; if the user cannot, we constrain them to 'do_now' and 'do_later'. We mask classifier outputs to enforce these constraints.

V. Evaluation

We conduct quantitative evaluation of TAACo on custom data obtained from real users. In this section, we describe evaluation data, baselines and metrics, and model training.

A. *Case Study*

To evaluate TAACo on real user data, we conducted an IRB approved case study about older adults' preferences across various household tasks through in-depth interviews with five people[3]. The resulting documentation $\mathcal{D}$ contains preferences of each person across 74-85 household tasks each, adding up to 173 distinct household tasks. All tasks have a unique action description, are associated with one of 60 high level activities, and utilize 265 objects and 14 locations in the household. The interview was based on scenarios, such as 'You asked the robot to help you prepare a meal... the robot is about to pour oil in a pan on the stove.' In context of each scenario, we asked four questions: 1) 'In the given scenario, how would you prefer Stretch to respond?', 2) 'Why? Please provide a reason why you picked the above.', 3) 'Would you respond differently in certain conditions and how?', and 4) 'In what other scenarios would you want Stretch to behave in this manner?'. We parse their responses into a set of datapoints $\{(\phi, \mathcal{E}_{user}, S_c)\}$, as shown in Table I and II. To ensure that tasks are relevant to the participants' daily lives and are not confined to a closed set, we allow the participants to skip or add tasks. We extract concepts from freeform text, to avoid restricting participants to a fixed concept vocabulary. Ultimately, for each persona, this process results in a set of tasks, along with the preferred task adaptation, and optionally an explanation and state constraints. We find that the resulting preferences vary significantly across personas (low average pairwise Cohen's Kappa of 0.23), and across tasks (76.5 rules for every 100 scenarios).

None of our participants had any significant constraints affecting their ability to perform household tasks. In order to validate how well our system accounts for capability constraints of the user or robot, we create dataset $\hat{\mathcal{D}}$, in which we simulate particular constraints on either the user's or the robot's capability, including 1) a persona sensitive to dust due to asthma, 2) a persona who cannot comfortably bend down due to back problems and arthritis, and 3) a robot with limited vertical reach, such as a stretch robot. Ultimately $\hat{\mathcal{D}}$ includes data for three personas, created by manually modifying preferred adaptations in $\mathcal{D}$.

[3]Gender and age ranges: M 80-85, M 70-75, F 70-75, F 65-70, F 75-80.

TABLE II. Examples of predictions and explanations generated by TAACo and baselines.

	Task t	State s	Ground Truth	TAACo	GPT	RuleBased
1	putting fruits in the blender to make a smoothie, Making a smoothie, {blender, kitchen counter, apple, banana, strawberry}, {kitchen}	user is in a rush & weekend	ϕ: no_action $\mathcal{E}$: putting fruits in the blender to make a smoothie is an action that a user might prefer doing themselves if they enjoy making food.	ϕ: no_action ✓ $\mathcal{E}$: putting fruits in the blender to make a smoothie is an action that a user might prefer doing themselves if they enjoy making food. ✓	ϕ: do_now ✗ $\mathcal{E}$: Making a smoothie is a/an activity which is falls under food preparation tasks. ✗	ϕ: do_now ✗ $\mathcal{E}$: - ✗
2	drilling holes in the wall to put up a coat hook, home decoration, {electric drill, hammer, screws}, {living room}	adverse weather & guests are present & weekend	ϕ: no_action $\mathcal{E}$: drilling holes in the wall to put up a coat hook is an action that can cause major damage or harm if done imprecisely, and electric drill is an object that can easily hurt someone without intending to	ϕ: no_action ✓ $\mathcal{E}$: drilling holes in the wall to put up a coat hook is an action that can cause major damage or harm if done imprecisely ✓	ϕ: no_action ✓ $\mathcal{E}$: Drilling holes in the wall to put up a coat hook is a/an action which can cause major damage or harm if done imprecisely ✓	ϕ: no_action ✓ $\mathcal{E}$: Object electric drill is involved ✓
3	arranging pots and pans in the kitchen shelves, organizing the kitchen, {pots, pans, kitchen shelves}, {kitchen}	user is asleep & weekend	ϕ: do_later $\mathcal{E}$: Arranging pots and pans in the kitchen shelves is an action that makes a lot of noise, and user is asleep	ϕ: no_action ✗ $\mathcal{E}$: Pot is an object which involves an open flame ✗	ϕ: do_now ✗ $\mathcal{E}$: Arranging pots and pans in the kitchen shelves is a/an action which is is very tiring ✗	ϕ: do_now ✗ $\mathcal{E}$: - ✗
4	watering house plants, maintaining house plants, {watering can, house plants}, {living room}	user is in a rush	ϕ: do_now $\mathcal{E}$: watering house plants is a task that a user might want to be carried out in a particular manner if they are particular about their house plants, and the user is in a rush	ϕ: do_now ✓ $\mathcal{E}$: user is in a rush ✓	ϕ: do_now ✓ $\mathcal{E}$: Watering house plants is an activity under maintaining house plants, which is a mundane chore that robots can assist with effectively ✗	ϕ: do_now ✓ $\mathcal{E}$: - ✗

B. Baselines

We benchmark TAACo against two baselines: end-to-end GPT-4 [43] and a rule-based system [17], and also compare against an oracle commonsense version with privileged access to concept matches. The GPT-4 baseline, uses a few-shot prompting approach. The prompt includes a full history of interactions including user responses and explanations from the training set, and asks for a response to a novel evaluation task. Inspired by end-user programming methods [17], the rule-based baseline creates explicit rules to predict desired behavior conditioned upon a component of the task and, optionally, a state variable, from explanations present in the training set. During inference, we use majority vote over applicable rules to predict the response. If no rules are applicable, we select a default response. Finally, the oracle commonsense version of our model corrects match scores in the intermediate task representation using ground truth explanations provided by the user, wherever available. This tests our personalization model's performance, assuming it has access to a perfect user-aligned scores, particularly for concepts deemed important by the user.

C. Metrics

We evaluate prediction and explanation accuracy of TAACo on our dataset. We measure **prediction accuracy** as the fraction of tasks where predicted label matches the ground truth label, and **explanation accuracy** as the fraction of tasks for which the top explanation offered by a model is a part of the ground truth set. For each model, we measure explanation accuracy only for tasks where ground truth explanations are available, and the model's prediction is correct, to avoid penalizing models for explaining their wrong answers incorrectly. Our explanation accuracy metric evaluates the top explanation alone, and not whether all ground truth explanations are generated, measuring whether the robot's response aligns with the user's expectation if it uses the top explanation.

D. Model Implementation and Training

All models, including TAACo as well as the baselines, are person-specific, and are trained and tested individually on each person's data using k-fold cross-validation. We evaluate performance across varying user feedback amounts by randomly selecting subsets of the training data. Starting with a base concept vocabulary, we expand it per person based on feedback, resulting in 23-27 concepts each. The state representation consists of 8 binary variables, combining all variables that participants used to define their preferences.

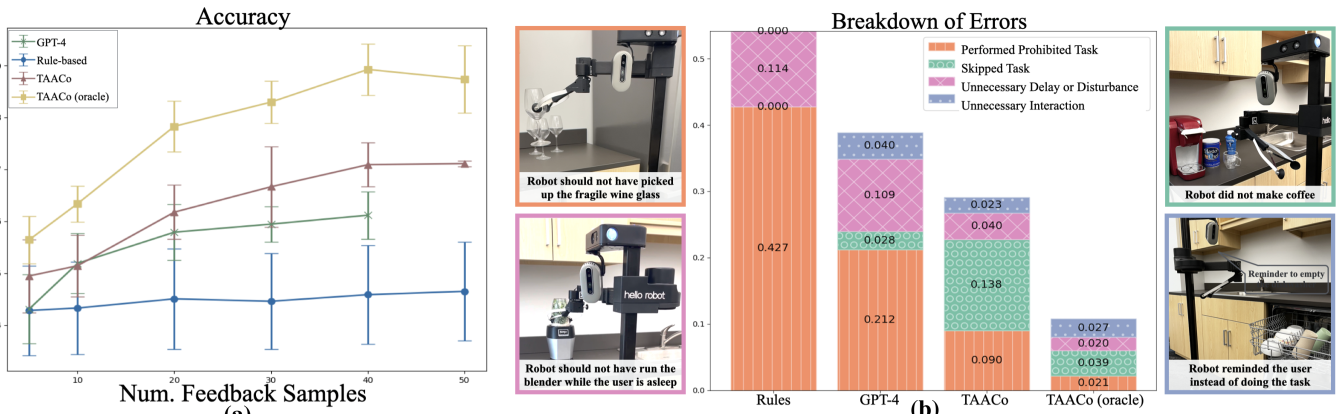
The Personalization Module uses 32-dimensional embeddings for type, concept, and score, creating 96-dimensional inputs for a 2-layer transformer encoder. We use Roberta sentence embeddings as the language encoder γ_{LLM}. The model, with 909K parameters, is trained for 1700 epochs with the Adam optimizer and a learning rate of 10^{-4}. We employ a linear combination of cross-entropy loss for predictions and auxiliary explanation loss for attention weights, with the latter given a weight of 20 against unit weight for the former. TAACo's Personalization module only takes on average 0.13s for inference, but Commonsense Module takes about 16s to predict match scores (for an average 29 θ^x-c^x pairs per task) using GPT-4, because queries can only be run sequentially. Batch-processing on a Llama3.1 model instead takes on average 2.1s.

VI. Results

In this section, we compare the prediction and explanation performance of TAACo against baselines and an oracle version of our model, and perform an analysis of the kinds of errors each model makes. We study how performance improves as more user feedback becomes available, by varying the number of user preference samples used in training, with each sample consisting of data pertaining to one task. We conduct all experiments on the original dataset $\mathcal{D}$, and evaluate TAACo's ability to handle capability constraints using dataset $\hat{\mathcal{D}}$ in Section VI-A. Finally, we empirically show the importance of using concepts and training on user-generated explanations.

A. Prediction Accuracy

Figure 3a shows a comparison of prediction accuracy across 10 to 40 labels obtained as user feedback. When little user



(a) (b)

Fig. 3. Comparison of our model against GPT and rule-based baselines, as well as against using an oracle version of commonsense, on (a) overall prediction accuracy over varying amounts of user feedback, and (b) a breakdown of the errors, with 40 feedback samples, with the each bars representing the total error rate and each color representing an error type, along with an example in a box of the same color

data is available, TAACo's performance is similar to GPT-4; TAACo outperforms both baselines as more samples become available. Owing to the vast commonsense knowledge, GPT-4 has a strong commonsense prior which is useful in guessing common preferences, but as some feedback becomes available, TAACo can learn nuances of the user's preferences better, despite GPT-4 having access to the exactly the same information.

Generalization to novel tasks is difficult because each new task has little direct overlap with those seen previously. This is evidenced in the rule-based baseline, which fails to improve with more examples because very few rules created from past examples are applicable to novel situations. Even with 40 training samples, it can only match learned rules for 17% responses, while the default 'do_now' happens to be correct for the remaining 29% correct tasks (e.g. row 4 in Table II).

TAACo achieves an accuracy of 0.71, while the oracle commonsense version achieves 0.89. This gap in performance results from misalignment between concept scores generated by the GPT-based commonsense module and that provided by the user. Despite the vast commonsense knowledge, GPT-4 can produce unusual or erroneous predictions, such as that the object 'pot' involves an open flame (row 3 in Table II). Also, there could be subjective differences between people's notions, such as whether being particular about food is applicable to mixing cake batter (rows 4,5 in Table I), causing the universal commonsense model to fail for some persona.

Even with the oracle commonsense, the model can fail due to lack of data or limited expressiveness of our representation. TAACo sometimes fails to learn a complex dependency on multiple variables which occur less frequently, such as task involving moving around, when the location is a tight space, and the user is present (row 6 in Table I). In rare cases, the preference may have never been seen in the train set. For instance, a person didn't want the robot to touch their pet, but neither of the two pet grooming tasks were seen in the training set. Other times, failures can occur when user preferences depend on details which can not expressed in our task representation, such as the difference in the role of the object stove in pouring oil in a pan on the stove, compared to turning on the stove (row 1,2 in Table I).

Finally, we evaluate TAACo on $\hat{\mathcal{D}}$, which accounts for capability constraints, with 40 feedback samples. We augment the prompt for the baseline GPT-4 model with the information about user context, and which adaptations should be disallowed if a constraint applies. TAACo achieves an average prediction accuracy of 0.708, compared to the GPT-4 baseline's accuracy of 0.642. Over the two personas with user capability constraints, we find that TAACo is able to identify whether the user can do the task with 91.4% accuracy.

B. Error Analysis

While prediction accuracy reflects overall model performance, different kinds of errors can carry different costs. To further investigate mistakes made by each model, we categorize errors, based on the their practical impact, as follows.

1) *Performed Prohibited Task* when the robot classified a task as *do_now* or *do_later*, but was supposed to be *no_action* or *remind*, implying that the robot would perform a task which the user meant to prevent it from executing.
2) *Skipped Task* when the robot misclassifies anything as *no_action*, risking the task remaining unfinished as the user might be expecting the robot to do it or remind them of it.
3) *Unnecessary Delay or Disturbance* when the robot misclassifies one of *do_now* and *do_later* as the other, disturbing the user or delaying the task unnecessarily.
4) *Unnecessary Interaction* when the robot wrongly predicts *remind*. While a system can be designed to allow the user to correct the robot, it involves interaction with the user.

Note that the four categories are mutually exclusive and exhaustively cover all errors. Figure 3b shows the distribution of errors made by each model, when trained on 40 samples, along with an example of each error type. The total height of each bar is the total error rate, and the colors represent the rate of each error type. Notably, most errors for the both baselines are of the least desirable type: *performed prohibited task*. The rule-based system's default 'do_now' action, helps prediction performance, due to the prevalence of 'do_now' actions, but causes majority of its errors to fall under this category. In contrast, TAACo only makes the *performed prohibited task* error 9% of the time, while a majority of its errors fall under *skipped task*. Overall, this implies that a robot using our model will be more conservative and err towards not acting when uncertain, rather than executing tasks which a user might

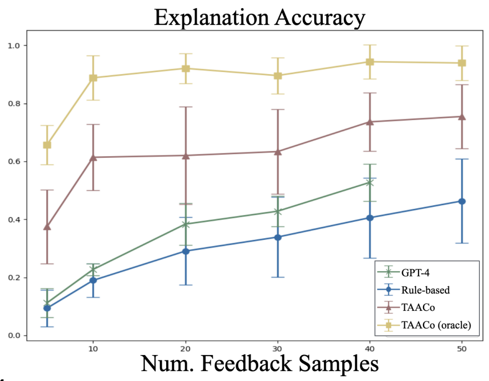


Fig. 4. Comparison of Explanation Accuracy against baselines.

not want it to do. Our method also shows a lower rate of unnecessary delay or disturbance and unnecessary interaction errors, compared to GPT-4, reducing smaller inconveniences such as interrupting the user by asking questions and making a ruckus, and adding to its conservative nature.

C. *Explanation Quality*

TAACo's explanations constitute a task component and associated concept, matching the ground truth explanation format, but our baselines' explanations do not. The rule-based baseline can use the applied rule to offer the task component as explanation, as seen in row 2 in Table II, but has no understanding of concepts, so we use a relaxed metric which deems an explanation to be correct if the right component is identified. For the GPT-4 baseline, despite prompting the system to generate an explanation in the given format, we were unable to fully constrain its responses (e.g. row 1 in Table II). So we employed GPT-4 to evaluate its own responses against ground truth through pairwise similarity queries.

Figure 4 shows a comparison of explanation accuracy of our model against both baselines and the oracle version, across varying levels of feedback. Our model outperforms both baselines, reaching an accuracy of 0.74 compared to 0.53 and 0.41 by GPT-4 and Rule-based conditions, respectively. The rule based baseline's inability to generalize rules from past feedback causes its explanation performance to be the lowest, despite a significantly relaxed evaluation metric.

D. *Ablations*

The core advancement of our model is the explicit use of abstract concepts, both for mediating reasoning and for explicit supervision from user feedback. Figure 5 summarizes performance benefits of both these elements on prediction and explanation accuracy, when trained on 40 feedback samples.

Our first hypothesis was that mediating reasoning through abstract concepts aids generalization by bootstrapping more relevant prior knowledge than directly using language embeddings. To test this, we compare against a 'No Concepts' version of our model which directly uses language embeddings of each component, instead of explicit concepts, as inputs to the Personalization Module, replacing the concept embedding and match score embedding. The no-concept model achieves only 58.5% prediction accuracy, compared to the 70.9% achieved by our model. Second, we hypothesized that explicitly training our model using explanations that the user provides can help

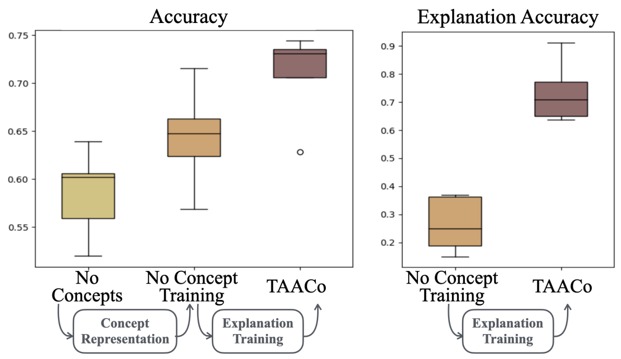


Fig. 5. Ablation results of our model, removing explanation training, and removing the use of concepts entirely.

it reason in a manner similar to the user, not only improving the model's explanation performance, but also its prediction performance. To test this, we compare our model against 'No Concept Training' version of our model, by removing the auxiliary explanation loss on the attention weights. This reduces the accuracy of explanations from 73.6% to 26.4%, and the prediction accuracy from 70.9% to 64.4%.

VII. Robot Demonstration

Finally, we show a proof-of-concept of our system on a stretch robot[4], showcasing personalization across a diverse array of tasks and associated concepts, as shown in the video and Figure 6. With our framework, the robot can accept feedback over a few tasks to personalize behavior, and generalize to a wider set of tasks across the household.

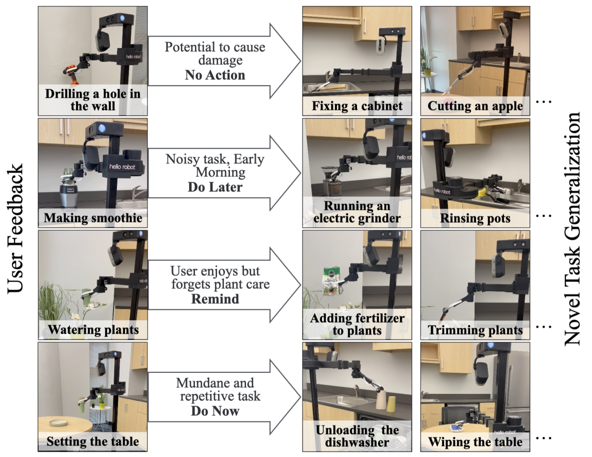

User Feedback
Drilling a hole in the wall
Potential to cause damage
No Action
Fixing a cabinet
Cutting an apple
Making smoothie
Noisy task, Early Morning
Do Later
Running an electric grinder
Rinsing pots
Watering plants
User enjoys but forgets plant care
Remind
Adding fertilizer to plants
Trimming plants
Setting the table
Mundane and repetitive task
Do Now
Unloading the dishwasher
Wiping the table
Novel Task Generalization

Fig. 6. Robot demo generalizing user feedback for different preferred task adaptations to novel household tasks.

VIII. Discussion and Future Work

In this work, we propose TAACo, which can help customize robot behavior to user preferences, conditioned on state context and capability constraints, by picking between four ways of assisting with a task: doing it now, doing it later, reminding the user about it, or not doing anything. Our core contribution is the use of abstract concept-based representation and training, which allows generalization to an open-set of household tasks and enables explainability. TAACo, by combining the generalization ability of an LLM with a tailored module to enable personalization, performs

[4]Full demo available at https://youtu.be/bFqubhycs-c

better than a baseline end-to-end LLM model, with today's leading LLMs. As LLMs improve in the future, both TAACo as well as the baseline would improve, but it is unclear which would benefit more.

For implementation on a real robot system, TAACo relies on access to 1) objects, action, locations and high-level activity information in the description for each task, 2) an NLP module which can parse user feedback in natural language to extract concepts, 3) a general robot policy or set of policies capable of performing the necessary tasks, and 4) perception modules which can infer user presence and activity. While TAACo relies on a functional robot policy, it can be extended to provide feedback to policy learning methods regarding which tasks the robot is more likely to perform. Future work can explore how such feedback can be incorporated in continual policy learning and measure the benefit of such feedback.

Our design of the TAACo framework also has some **limitations**, opening avenues for future work. First, our input task representation includes a textual action and activity description, and a set of objects and locations, which cannot capture details, such as the role of an object in the given task, which might be pertinent to some preferences. Second, for some individuals or environments, the notion of component-concept alignment might diverge from the norm. Since we do not customize the commonsense module, we cannot model such preferences. Future work should explore ideas of modeling change in user preferences, lifelong learning in context of this problem, as well as evaluate TAACo more extensively over more diverse domains.

References

[1] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "Vima: General robot manipulation with multimodal prompts," in *ICML*, 2023.

[2] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," pp. 785–799, PMLR, 2023.

[3] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE RAL*, 2023.

[4] –, "Learning to learn faster from human feedback with language model predictive control," 2024.

[5] M. Patel and S. Chernova, "Proactive robot assistance via spatio-temporal object modeling," vol. 205, pp. 881–891, PMLR, 2023.

[6] M. Patel, A. G. Prakash, and S. Chernova, "Predicting routine object usage for proactive robot assistance," vol. 229, PMLR, 2023.

[7] E. V. Mascaro, D. Sliwowski, and D. Lee, "HOI4ABOT: Human-object interaction anticipation for human intention reading collaborative roBOTs," in *7th Annual*, 2023.

[8] I. Kapelyukh and E. Johns, "My house, my rules: Learning tidying preferences with graph neural networks," pp. 740–749, PMLR, 2022.

[9] V. Jain, Y. Lin, E. Undersander, Y. Bisk, and A. Rai, "Transformers are adaptable task planners," pp. 1011–1037, PMLR, 2023.

[10] A. Peng, A. Netanyahu, M. K. Ho, T. Shu, A. Bobu, J. Shah, and P. Agrawal, "Diagnosis, feedback, adaptation: A human-in-the-loop framework for test-time policy adaptation," in *ICML*, 2023.

[11] L. Yuan, X. Gao, Z. Zheng, M. Edmonds, Y. N. Wu, F. Rossano, H. Lu, Y. Zhu, and S.-C. Zhu, "In situ bidirectional human-robot value alignment," *Science robotics*, vol. 7, 2022.

[12] F. Ranz, V. Hummel, and W. Sihn, "Capability-based task allocation in human-robot collaboration," *Procedia Manufacturing*, vol. 9, 2017.

[13] A. A. Malik and A. Bilberg, "Complexity-based task allocation in human-robot collaborative assembly," *Industrial Robot: the international journal of robotics research and application*, vol. 46, 2019.

[14] N. Gjeldum, A. Aljinovic, M. Crnjac Zizic, and M. Mladineo, "Collaborative robot task allocation on an assembly line using the decision support system," *International Journal of Computer Integrated Manufacturing*, vol. 35, 2022.

[15] M. D. Zhao, R. Simmons, and H. Admoni, "Learning human contribution preferences in collaborative human-robot tasks," in *CORL*, 2023.

[16] C.-A. Smarr, T. L. Mitzner, J. M. Beer, A. Prakash, T. L. Chen, C. C. Kemp, and W. A. Rogers, "Domestic robots for older adults: attitudes, preferences, and potential," *IJSC*, vol. 6, 2014.

[17] J. Saunders, D. S. Syrdal, K. L. Koay, N. Burke, and K. Dautenhahn, ""teach me–show me"—end-user personalization of a smart home and companion robot," *HMS*, vol. 46, 2015.

[18] N. Leonardi, M. Manca, F. Paternò, and C. Santoro, "Trigger-action programming for personalising humanoid robot behaviour," in *CHI*, 2019.

[19] S. Zhuang and D. Hadfield-Menell, "Consequences of misaligned ai," *NeurIPS*, vol. 33, 2020.

[20] J. Skalse, N. Howe, D. Krasheninnikov, and D. Krueger, "Defining and characterizing reward gaming," in *NeurIPS*, vol. 35, 2022.

[21] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh, "No, to the right: Online language corrections for robotic manipulation via shared autonomy," in *HRI*, 2023.

[22] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox, "Correcting robot plans with natural language feedback," *arXiv*, 2022.

[23] Y. Cui, Q. Zhang, B. Knox, A. Allievi, P. Stone, and S. Niekum, "The empathic framework for task learning from implicit human feedback," vol. 155, PMLR, 2021.

[24] K. Ramachandruni, M. Zuo, and S. Chernova, "Consor: A context-aware semantic object rearrangement framework for partially arranged scenes," in *IROS*, IEEE, 2023.

[25] D. Lindner, R. Shah, P. Abbeel, and A. Dragan, "Learning what to do by simulating the past," *arXiv*, 2021.

[26] T. Munzer, M. Toussaint, and M. Lopes, "Preference learning on the execution of collaborative human-robot tasks," in *ICRA*, IEEE, 2017.

[27] M. Zhao, R. Simmons, and H. Admoni, "Coordination with humans via strategy matching," in *IROS*, IEEE, 2022.

[28] H. Nemlekar, N. Dhanaraj, A. Guan, S. K. Gupta, and S. Nikolaidis, "Transfer learning of human preferences for proactive robot assistance in assembly tasks," in *ACM/IEEE HRI*, 2023.

[29] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *NeurIPS*, vol. 29, 2016.

[30] Y. Du, S. Tiomkin, E. Kiciman, D. Polani, P. Abbeel, and A. Dragan, "Ave: Assistance via empowerment," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4560–4571, 2020.

[31] B. Quartey, E. Rosen, S. Tellex, and G. Konidaris, "Verifiably following complex robot instructions with foundation models," *arXiv*, 2024.

[32] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *IEEE/CVF CVPR*, pp. 2998–3009, 2023.

[33] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," in *ICRA*, IEEE, 2023.

[34] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *ICRA*, IEEE, 2023.

[35] D. Sobrín-Hidalgo, M. A. González-Santamarta, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, and V. Matellán-Olivera, "Explaining autonomy: Enhancing human-robot interaction through explanation generation with large language models," *arXiv*, 2024.

[36] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "Tidybot: Personalized robot assistance with large language models," *arXiv*, 2023.

[37] B. Zhang and H. Soh, "Large language models as zero-shot human models for human-robot interaction," in *IROS*, IEEE, 2023.

[38] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv*, 2022.

[39] K. E. C. Booth, T. T. Tran, G. Nejat, and J. C. Beck, "Mixed-integer and constraint programming techniques for mobile robot task planning," *IEEE RAL*, vol. 1, 2016.

[40] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach.," in *RSS*, vol. 12, 2016.

[41] D. Das, S. Chernova, and B. Kim, "State2explanation: Concept-based explanations to benefit agent learning and user understanding," 2023.

[42] R. Zabounidis, J. Campbell, S. Stepputtis, D. Hughes, and K. P. Sycara, "Concept learning for interpretable multi-agent reinforcement learning," pp. 1828–1837, PMLR, 2023.

[43] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, *et al.*, "Gpt-4 technical report," *arXiv*, 2023.