Grounding English Commands to Reward Functions

James MacGlashan* Monica Babeş-Vroman+ Marie desJardins‡ Michael Littman* Smaranda Muresan§
Shawn Squire‡ Stefanie Tellex* Dilip Arumugam* Lei Yang†

***Abstract*—As intelligent robots become more prevalent, methods to make interaction with the robots more accessible are increasingly important. Communicating the tasks that a person wants the robot to carry out via natural language, and training the robot to ground the natural language through demonstration, are especially appealing approaches for interaction, since they do not require a technical background. However, existing approaches map natural language commands to robot *command languages* that directly express the sequence of actions the robot should execute. This sequence is often specific to a particular situation and does not generalize to new situations. To address this problem, we present a system that grounds natural language commands into *reward functions* using demonstrations of different natural language commands being carried out in the environment. Because language is grounded to reward functions, rather than explicit actions that the robot can perform, commands can be high-level, carried out in novel environments autonomously, and even transferred to other robots with different action spaces. We demonstrate that our learned model can be both generalized to novel environments and transferred to a robot with a different action space than the action space used during training.**

I. Introduction

For robots and other intelligent agents to be useful to the general public, they need to be able to autonomously carry out complex tasks. However, it is equally important for humans to be able to communicate a desired complex task to an agent, ideally using natural language commands instead of a formal machine-oriented task representation. In this work, we present a method for learning how to ground natural language commands into high-level reward functions from demonstrations of the commands being carried out.

Previous approaches to interpreting language for robots have mapped between natural language and specific planning languages or feature representations that directly describe the sequence of actions the robot should perform [14, 21, 16, 15, 6]. In effect, these approaches enable teleoperation though language. However, in domains where a robot is expected to assist a human, requiring the human to uniquely specify the action sequence necessary to complete a task is an undesirable burden, especially if the environment changes over use cases (entailing different action sequences to be used for the same task), or if stochasticity in the environment or actions can cause a specified action sequence to fail to complete the task. Ideally, a human would specify the task to be completed with language, and the robot would then use a planner to be creative in its solutions to the task, autonomously resolving any unforeseen outcomes during execution. For example, the command "bring me a cup of coffee" should give rise to a goal that motivates the robot to choose steps for getting to—or making, if necessary—the coffee and bringing it to the user, without the user specifying any of those details in the command. Another advantage of this task-based grounding is that a language model grounded to tasks can be easily transferred to different robots with different action sets, since the agent will use a planner to determine the correct actions.

An alternative way to learn groundings to reward functions would be to train a semantic language model with a dataset consisting of pairs of natural language commands and the corresponding reward function descriptions. However, providing such a dataset would require a trainer to have a technical understanding of the task and state representation of the robot. Our more user-friendly approach to training is for the user to develop a dataset of commands paired with demonstrations of the command being carried out.

Our work has two primary contributions: first, a generative model of tasks, behavior, and language; second, a weakly supervised learning algorithm that uses our generative model and a version of inverse reinforcement learning (IRL) [19] to learn mappings from language to latent reward functions from a dataset consisting of natural language commands and demonstrations of the commands being carried out.

Our approach is related to the approach described by Howard et al. [11]. However, our work focuses on grounding language to tasks best represented by Markov decision process (MDP) reward functions and solved by planning under uncertainty, whereas Howard et al. focus on grounding language to planning constraints for motion planning.

We empirically validate our approach using a pick-and-place domain in which a robot may be tasked with going to different rooms or taking certain objects to different rooms, using both a simulated version of the domain and as a demonstration on a physical robot. Our simulated results demonstrate that our approach transfers to new environments, significantly outperforming baseline methods that ground language to actions. Our task-grounding model performs well both when the test environments are the same as the training environments and when the environments are novel. In contrast, an action-grounding model is only able to perform well in the known environments and fails in novel environments. We also demonstrate that the learned model that was trained on simulated data can be transferred to a physical robot with a different action space that manipulates blocks in the physical world. The code for

* Computer Science Department, Brown University
+ Computer Science Department, Rutgers University
‡ Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
§ Department of Computer Science, Columbia University
† School of Software, Tsinghua University

this work is freely available online.[1]

II. Object-oriented Markov Decision Processes

To represent tasks, we make use of the *Object-oriented Markov Decision Process* (OO-MDP) formalism [8] because it is well suited to represent object-based knowledge for planning under uncertainty that can transfer across different environments. OO-MDPs extend the conventional MDP formalism by providing a rich factored state representation that describes the objects in the environment.

MDPs are defined by a five-tuple: ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, $\mathcal{F}$), where $\mathcal{S}$ is a set of states of the world; $\mathcal{A}$ is a set of actions that the agent can take; $\mathcal{T}$ describes the transition dynamics, which specify the probability of the agent transitioning to each state given the current state and the action taken; $\mathcal{R}$ is a function specifying the reward received by the agent for each transition; and $\mathcal{F}$ is a set of terminal states that cause action to cease once reached. The goal of planning in an MDP is to find a *policy*—a mapping from states to actions—that maximizes the expected discounted cumulative reward.

An OO-MDP extends the classic MDP formalism by including a set of *object classes*, each defined by a set of attributes, and a set of *propositional functions* whose parameters are typed to object classes. A state in an OO-MDP is a set of objects that each belong to one of the possible object classes; each object has its own state, which is represented as a value assignment to the attributes of its associated object class. The propositional functions defined in the OO-MDP are functions of the object states being evaluated. For example, consider a blocks world in which block objects are defined by their position in the world. The propositional function evaluation $\texttt{on}(b_1, b_2)$, where b_1 and b_2 are block objects in a state, returns true when the position of b_1 is adjacent and above the position of b_2 and false otherwise.

The propositional functions enable an OO-MDP to provide high-level symbolic information about states that are inherently non-symbolic (e.g., spatial or continuous), which are often needed in robotics domains. Because OO-MDP states are defined by a set of objects, different states in the same OO-MDP can also represent different environments by changing which objects are present. Since propositional functions operate on objects, the propositional functions generalize across environments. In this work, we use OO-MDP propositional functions to define abstract task definitions, which enables environment-independent symbolic tasks useful for language grounding to be defined for a variety of different kinds of state spaces that may not be symbolic themselves.

III. Execution and Learning

In this section, we describe the overall execution and learning process of our approach. In the next section, we will detail our generative model of tasks, behavior, and language that enables these processes. As a running example, consider the initial state of our *Cleanup World* domain

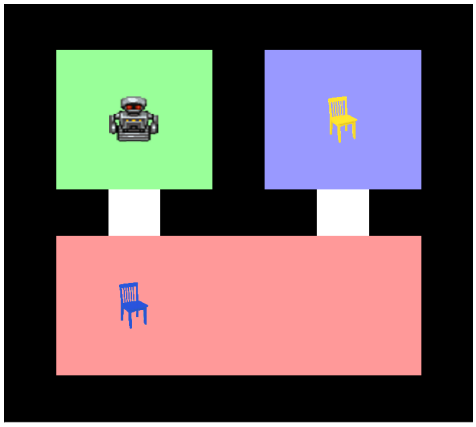


Fig. 1: An example three-room layout with one robot and two chairs. The object references for the red (lower), green (upper left), and blue (upper right) rooms are r_1, r_2, and r_3, respectively. The references for the yellow (upper) and blue (lower) chair are c_1 and c_2.

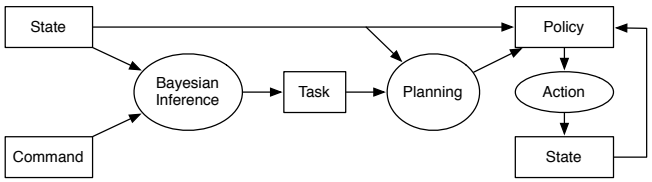
shown in Figure 1. Cleanup World (inspired by Sokoban [12]) is a 2D grid world of various rooms connected by open doors. Rooms can also contain "blocks" that can be moved around. The robot moves using north-south-east-west actions. Moving into a location containing a block causes the block to move in the direction the agent is moving. If a wall or other item is in its immediate path, neither the agent nor the block moves. Cleanup World is represented as an OO-MDP consisting of four object classes: ROBOT, BLOCK, ROOM, and DOORWAY. The propositional functions defined for the OO-MDP include $\texttt{robotInRoom}(\texttt{ROBOT}, \texttt{ROOM})$, and $\texttt{blockInRoom}(\texttt{BLOCK}, \texttt{ROOM})$, as well as propositional functions to indicate the color and type of rooms and blocks (e.g., $\texttt{roomIsOrange}(\texttt{ROOM})$, $\texttt{blockIsStar}(\texttt{BLOCK})$).

Figure 2a shows a flow chart of the execution process. To illustrate the steps of this process, considering the command "take the yellow chair to the red room" being given in the state shown in Figure 1. Using Bayesian inference with the learned model, the most likely task for the state shown in Figure2a is the goal condition $\texttt{blockInRoom}(b_1, r_1)$, because the command uses words for taking an object to a location, and because the selected objects satisfy the properties $\texttt{roomIsRed}(r_1) \wedge \texttt{blockIsYellow}(b_1) \wedge \texttt{blockIsChair}(b_1)$, which would have a high probability of generating the English words in the command. Once the most likely task has been inferred, a standard plan and control problem is instantiated, in which the robot uses a planning algorithm to generate a policy and follows the policy in a closed loop fashion.[2] Our approach is independent of the planning algorithm: it can use any "off-the-shelf" planning algorithm appropriate for the domain, allowing our approach to apply to a large variety of robotics domains as long as enough of the state is observable to perform task grounding.

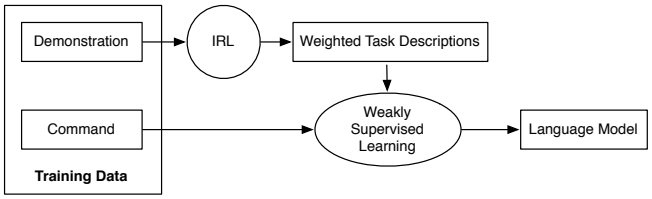
Figure 2b shows a flow chart of the learning process for

[1] https://github.com/jmacglashan/commandsToTasks

[2] The policy does not have to be complete and can invoke replanning if it is more efficient to do so.



(a) A flow chart of command execution.



(b) A flow chart of the training process.

Fig. 2: Flow charts for command execution and learning.

taking a dataset of demonstrations paired with commands and learning the parameters of a language model that ground language to latent task descriptions. Learning requires the set of OO-MDP propositional functions to which language is ground to be known in advance, but it does not require the natural language to be known in advance. Consider the previous command "take the yellow chair to the red room" in the role of a training instance, paired with a demonstration that starts in the initial state shown in Figure 1 and then follows the action sequence $SSSEEEENNWNNESSS$, where N, S, E, W stand for north, south, east, and west, respectively. From this demonstration, inverse reinforcement learning (IRL) infers the probability that each possible task in the environment of the demonstration was the task being completed by the demonstration. The task with the goal $\texttt{blockInRoom}(b_1, r_1)$ will have a very high probability compared to other tasks. This task has many possible descriptions based on the properties of the objects b_1 and b_2 to which language could be ground. In particular, one such task description is $\texttt{blockInRoom}(b_1, r_1)\texttt{roomIsRed}(r_1) \wedge \texttt{blockIsYellow}(b_1) \wedge \texttt{blockIsChair}(b_1)$. The set of all possible task descriptions for all tasks, weighted by the probability of their task being the intended one, form a set of weakly supervised labels for the training command and are used with a weakly supervised learning algorithm to train the language model parameters.

Our approach can easily incorporate different semantic language models. We investigate two language models—a bag-of-words (BoW) mixture model and the IBM Model 2 (IBM2) used in statistical machine translation [4, 5]—and use expectation maximization [7] for training them with the weakly labeled commands. While other language models that incorporate grammatical knowledge might yield better performance and robustness, an advantage of these grammar-free models is that they do not require any additional corpus training and can be used with our task and behavior model as is.

For these language models, semantic parsing through inference on a trained model was very fast, because the set of semantic interpretations was small compared to translating to another full natural language, for which IBM2 is typically used. The more demanding process of the command execution is planning, which will increase in complexity as the environment becomes more complex. However, since different planning algorithms may be used, this approach will scale to more complex environments as planning algorithms scale.

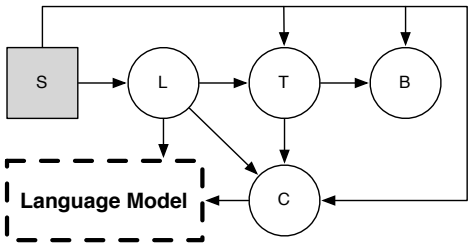


Fig. 3: The generative task model, with arrows indicating conditional probabilities. S is the current state and is an input to the model. L is the lifted task; T is the grounded task; B is the behavior of the agent; and C is the object-binding constraints. The language model is a parameter of our overall generative model.

Given the weakly labeled task descriptions, training the language models was also fast using the expectation maximization algorithm [7] (see Section V-C). The primary computational cost lies in performing IRL for each demonstration. In our domain, performing IRL on a demonstration requires planning for all the finite possible tasks in the environment. As with command execution, this cost will scale to more complex environments as planning algorithms scale. In our tests, the number of possible tasks in the environment was not prohibitive. In domains with a very large number of tasks per environment, approximate inference or a scaffolding procedure in which the trainer iteratively trains the robot with simple environments may be beneficial. We leave this investigation for future work.

IV. The Generative Model

Our generative model of tasks, behavior, and language used for learning and command execution is shown in Figure 3. It consists of an input initial state (S), a set of lifted tasks (L), a set of grounded tasks (T), a set of object-binding constraints (C), a set of behavior demonstrations/trajectories (B), and a language model that produces natural language commands and is dependent on the lifted task and binding constraints. A lifted task specifies the kinds of problems the agent can solve with propositional functions, but leaves the OO-MDP objects on which it operates as free variables; a grounded task binds a lifted task's free variables to specific objects in the world, thereby producing a well defined decision-making problem; object-binding constraints specify the set of possible properties of the grounded task's bound objects that a user might refer to in language; and a behavior trajectory is a sequence of states and actions taken by the agent.

A. Lifted Tasks

The set of possible lifted task definitions is provided by a designer and represents the kinds of tasks an agent could conceivably carry out in an environment. Specifically, each lifted task is a factored reward function defined as a logical expression of OO-MDP propositional functions with the parameters of the propositional functions set to free variables. In our example environment (Figure 1), a reward function for the task of taking a "block" object (e.g., chair) to a room is:

$$\mathcal{R}(s,a,s') = \begin{cases} 1 & \text{if } \texttt{blockInRoom}^{s'}(?b,?r) \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

where s' is the outcome state for the agent after taking action a in state s, $\texttt{blockInRoom}^{s'}$ is a propositional function evaluated in state s', $?b$ is a free variable that can be grounded to any movable block object in the environment, and $?r$ is a free variable that can be grounded to any room in the environment. If a task is goal-directed, then the reward function will also be paired with a similar set of termination conditions.

The prior probability distribution on the set of lifted tasks is assumed to be uniform over the set of lifted tasks that are *permissible* in an input state. A lifted task is permissible in an input state if there exists at least one object of each object class required by the free variables in the lifted task. For example, if the input state has no block objects present, then lifted task 1 is *not* permissible in the state.

B. Grounded Tasks

A grounded task (T) is dependent on the lifted task and input state and is an assignment of objects in the input state to each of the free variables in the lifted task. For example, given the input state shown in Figure 1, an object assignment to lifted task 1 that represents the task of the robot taking the yellow chair to the red room is $?b := c_1$ and $?r := r_1$. The probability distribution of grounded tasks is uniform over the set of possible variable assignments for the lifted task. In goal-directed tasks, only tasks that are not satisfied in the initial state are considered. Because our task model starts with lifted tasks that are defined with free variables and then finds all possible groundings in the current input state, it generalizes to environments with any number of objects present.

C. Object-Binding Constraints

If language was only dependent on the lifted task, it would be impossible for the agent to determine which of the possible grounded tasks for an input state was the intended task from a command. Object-binding constraints (C) are logical expressions that extend a lifted task and on which language is dependent, enabling the agent to resolve grounding ambiguity. For example, the command "take the yellow chair to the red room" indicates that a grounding for lifted task 1 should satisfy the constraints $\texttt{isYellow}(?b) \wedge \texttt{isRed}(?r)$.

The probability distribution of object constraints given a grounded task, lifted task, and input state is uniform across the set of possible additional logical constraints that are true in the input state for the given variable assignments in the grounded task. For example, if $?b$ is assigned to c_1 in the grounded task, $\texttt{isYellow}(?b)$, is a possible constraint since it is true in the initial state for c_1, but $\texttt{isBlue}(?b)$ is not.

D. Behavior

A *behavior trajectory* is a sequence of state–action pairs that the agent can experience from an input state and is conditionally dependent on the grounded task and input state. The conditional probability of the behavior, given the task, is formulated by treating each action selection in each state of the trajectory as an independent event and by defining the action-selection probability distribution as a noisy version of the policy computed by an "off-the-shelf" planning algorithm. Following maximum likelihood inverse reinforcement learning [1], we use a Boltzmann distribution over the optimal Q-values as the noisy policy. The probability of any behavior trajectory b (of length N) given task t is defined as:

$$\Pr(b|t) = \prod_{i}^{N} \pi_t(s_i, a_i), \qquad (2)$$

where (s_i, a_i) is the ith state–action pair in behavior trajectory b and $\pi_t(s_i, a_i)$ is the noisy policy distribution under task t (the probability of taking action a_i in state s_i when the task is task t).

For goal-directed tasks, we add a virtual "terminate" action to the MDP that must be executed to actually terminate the task; this virtual terminate action is added to the end of observed trajectories. The inclusion of a terminating action more strongly differentiates between tasks where the optimal trajectory for one task is a subsequence of the optimal trajectory for another. For example, an optimal trajectory for going to the red room is a subsequence of an optimal trajectory for going to the blue room (in Figure 1), but since it is not expected for the agent to terminate in the red room if the task is to go the blue room, terminating in the red room makes it more likely for the trajectory to be generated by a "go to the red room" task.

V. Language Models

We investigated two grammar-free language models for use with our task model: a BoW model and IBM Model 2.

A. Bag-of-Words

In the bag-of-words (BoW) model, commands are treated as an unordered bag of words generated from a mixture model of semantic components, similar to previous topic modeling approaches [17]. The semantic components, given a lifted task and object-binding constraints, include (1) the set of propositional function names included in the task and binding-constraint expressions, (2) the object class names of the arguments of the propositional functions, and (3) a constant symbol ($\#$) that acts as a catchall for words not explicitly related to a semantic component. For example, lifted task 1 coupled with the constraint $\texttt{isRed}(?r)$ would have the semantic components $\texttt{blockInRoom}, \texttt{isRed}, \texttt{BLOCK}, \texttt{ROOM}$, and $\#$. The probability

that any semantic component is selected to generate a word is proportional to its frequency of appearance in the task and binding-constraint expressions.

The parameters of the BoW model are the conditional probabilities that any word will be generated from a given semantic component. Therefore, the probability of a command (e) given a lifted task (l) and object-binding constraints (c) is

$$\Pr(e|l,c) = \prod_{w \in e} \left[\sum_{v} \Pr(v|l,c)\theta_{vw} \right]^{K(w,e)}, \quad (3)$$

where θ_{vw} is a parameter specifying the conditional probability that the natural language word w is generated given semantic component v, and $K(w,e)$ is the number of times that word w appears in command e.

B. IBM Model 2

IBM Model 2 (IBM2) [4, 5] is a word-based statistical machine-translation model. In statistical machine translation, the task is to translate a sentence from a source language f (e.g., French) to a target language e (e.g., English). Our task corresponds to the problem of translating from an English command e to its corresponding machine-language command m.

We integrate IBM2 into our task and behavior model by using the lifted task and object-binding constraints to deterministically generate a machine-language command m and then generate the natural language command e from m in the standard IBM2 fashion. A machine-language command (m) is generated from a lifted task (l) and object-binding constraints (c) by first adding each semantic component in l to m in the order that the components appear. Next, the same is done for the components of c. For example, the lifted task in Equation 1 and object-binding constraint $\texttt{isRed}(?r)$ would generate the machine-language command "# blockInRoom block room isRed room," where # is the constant symbol that IBM2 always assumes to be present at the start of an expression. The probability of a natural language command (e) given the machine-language command (m) is defined as:

$$\Pr(\mathbf{e}|\mathbf{m}) = \eta(n_e|n_m) \sum_{\mathbf{a}} \prod_{j}^{n_e} q(a_j|j,n_m,n_e) r(e_j|m_{a_j}), \quad (4)$$

where $\eta(n_e|n_m)$ is the parameter specifying the probability that a machine-language command of length n_m would generate a natural language command of length n_e; $\mathbf{a}$ is a possible alignment from natural language words to machine-language words; $q(a_j|j,n_m,n_e)$ is the alignment parameter specifying the probability that the natural language word in position j would be aligned with the machine-language word in position a_j for a machine-language command of length n_m and natural language command of length n_e; and $r(e_j|m_{a_j})$ is the translation parameter specifying the probability that natural language word e_j would be generated from machine-language word m_{a_j}. The number of alignments ($\mathbf{a}$) is typically very large, so in practice we estimate the value using sampling.

C. Learning Language Model Parameters

Given a dataset of trajectory demonstration and natural language command pairs, the language model parameters can be learned through weakly supervised learning.

When the task and behavior model is paired with the BoW language model, we use a standard Bayesian Network expectation maximization (EM) algorithm [7] to iteratively update the BoW model's word generation parameters.

In classic IBM2 parameter learning, EM is used on a training dataset consisting of pairs of natural language expressions (e.g., French and English), often with a "bake-in" period, during which only translation parameters are updated, followed by a normal learning phase during which translation parameters and alignment parameters are updated. We follow the same approach here, except in our case, we have pairs of machine language commands and English commands, and the contribution of the expectation of machine language commands is weighted by the probability of being generated given the demonstration (see Section III). The probability of any machine-language command (m) given a trajectory (b) and initial state (s) is:

$$\Pr(m|s,b) = \frac{\sum_{l,t,c} \Pr(l|s) \Pr(t|s,l) \Pr(b|s,t) \Pr(c|s,l,t) \Pr(m|l,c)}{\sum_{l,t} \Pr(l|s) \Pr(t|s,l) \Pr(b|s,t)}. \quad (5)$$

A similar approach could be used for training the parameters of other language models.

VI. Experimental Results

To empirically validate our approach, we collected example language for the "Cleanup World" from two different sources: an Amazon Mechanical Turk (AMT) user study and an "expert" who is an author of this paper. Different experiments were run for the AMT language source and the expert source. In the AMT study, we asked users to provide example commands for a smaller range of tasks, but used more abstract object representations and unusual color choices to elicit more variety in the command descriptions. In the expert study, we evaluated a larger range of tasks and environments but with fairly clean expressions of the language. The details of the various experiments run with each study are explained below. We also show how we can take the language model learned from the AMT study, recreate the Cleanup World domain on an actual mobile robot, and successfully transfer the learned model so that the robot correctly interprets the commands, even though the robot has a different action space than the simulated robot used for training.

In both studies, the Boltzmann policy parameter τ was set to 0.005. Since we knew the task that the training commands were meant to describe, performance was measured using leave-one-out (LOO) cross validation on the collected training examples; a prediction was considered correct if the interpretation of the reward function resulted in behavior that achieved the actual goal.

For a baseline, we compared to a model that grounds language to actions rather than tasks. Because the order of

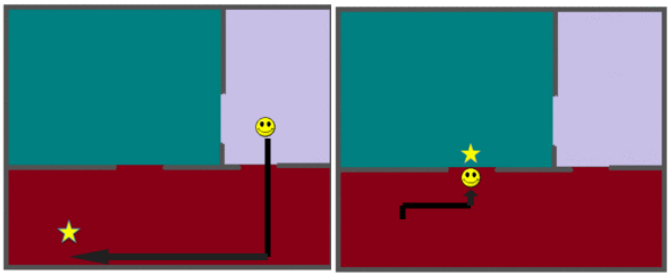


Fig. 4: An example task to take a star to a room shown to users on AMT. The left frame shows the initial state and the path to the star; the right frame shows the path from there to the end state. (In the actual study, users were shown a single animated image without arrows.)

the actions is important, we always paired the baseline with the IBM2 language model. When training the IBM2 model, the action sequence in the demonstration is converted into a machine-language sentence in which each word is the name of the action. Note that the action-grounding learning problem is in some sense easier than training the task model, since the action sequence is always directly observed in the demonstration, whereas in our task grounding model, IRL is needed to perform weakly supervised learning. Inferring an action sequence from a command would generally require searching the action sequence space and finding the most likely sequence from the command or at least using it to bias a planning algorithm's search, similar to the work of Chen and Mooney [6]. However, we simplify this inference problem greatly by allowing the action-grounding model to exploit our task model. Specifically, when inferring an action sequence for a command, the only valid action sequences permitted are those that solve one of the possible grounded tasks; the baseline always chooses the most likely of the possible sequences.

A. AMT Dataset

In the AMT study, we considered two different lifted tasks: the robot going to a specific room and the robot moving a block to a specific room. To collect natural language instructions for different grounded versions of these tasks, we presented Turkers on AMT animated images showing either the robot moving to a room of a specific color or moving a star block to a room of a specific color. An example image from the animation is shown in Figure 4. To prevent contamination of the commands we received, we never provided users with any example commands.

After removing sentences that did not follow the instructions or were provided by users who did not understand the labeling task that we were asking them to perform, we obtained a dataset of 240 instances. In a final system used by individual users iteratively training a robot, they would understand the task they were trying to train the robot to follow, so we do not expect similar command removal to be necessary.

The goal of our work is to be able to give autonomous agents high-level commands that leave the details of how to complete the task as a problem for the agent to solve. However, most of the natural language commands we received included some high-level details of the path the robot followed (e.g., subgoals), rather than only describing the task goal. For example, one such command was "go through center opening into beige enclosure and get behind star and push it into opening of orange enclosure." Although this data is interesting because it tests our model's performance on language that it was not intended to model, a dataset that better reflects the problem we were trying to solve is also useful for comparison. Therefore, in addition to this source dataset, we created a simplified version that omits the text that is extraneous to the task description. For instance, the previous example became: "go through center opening and get behind star and push it into opening of orange enclosure." The average command lengths (in words) were 13.57 and 8.87 in the original and simplified datasets, respectively. We tested results on both the original dataset and this simplified version.

In the training data we gathered from users, the room layout was always the same; however, one of the advantages of our approach is that commands can generalize to novel environments. To demonstrate the ability for our approach to generalize to new environments, we created a third dataset in which we used LOO cross validation for the language on the original (unmodified) dataset and tested each command on three different novel environments. Note that these different environments were not merely a different spatial arrangement of the rooms compared to the training dataset; the environments also had different initial conditions for the agent not seen in the training environment, which results in a different set of propositional functions being true in the initial state. As a consequence, our task model could perform worse if, during training, it overfit the object-binding constraints to inappropriate features. We refer to this dataset as the *novel environment* (NE) dataset.

The LOO accuracy for each grounding model (task and action) and the paired language model (BoW and IBM2) on each variant of the dataset is shown in Table I. Note for comparison that if the agent randomly selected a permissible grounded task, it would achieve an expected accuracy of 37.5% in the simplified and original dataset and 33.3% accuracy in the NE dataset.

On the simplified dataset, task grounding with both BoW and IBM2 performed well and comparably (according to a chi-squared test, BoW and IBM2 are not statistically different; $p = 0.54$). However, while action grounding was able to perform better than randomly guessing, it did not perform as well as BoW or IBM2; the difference was statistically significant ($p < 0.001$ on a chi-squared test with all three groups).

On the original dataset, task grounding performance dropped for both language models, although BoW's drop in performance was much greater than IBM2's drop in performance and of the two, only BoW's drop in performance was statistically significant ($p < 0.01$ and $p = 0.07$ for BoW and IBM2's decrease in performance, respectively). The action grounding baseline performed identically to its performance

Grounding Model	Simplified Dataset	Original Dataset	NE Dataset
Task (BoW)	**83.75%**	53.75%	51.80%
Task (IBM2)	81.25%	**74.16%**	**73.33%**
Action (IBM2)	69.58%	69.58%	21.94%

TABLE I: LOO accuracy for the AMT datasets.

Grounding Model	Expert	Expert-MD
Task (BoW)	48.30%	52.28%
Task (IBM2)	**65.25%**	**79.73%**
Action (IBM2)	49.15%	49.67%

TABLE II: LOO accuracy for the expert datasets.

on the simplified dataset and is reasonably comparable to task grounding using IBM2.

On the NE dataset, both task models were able to successfully generalize from the training data to the novel environments, achieving nearly the same performance as its performance on the original dataset for the corresponding language model. In contrast, the action grounding baseline completely failed to generalize from its training data, achieving a performance worse than what would be expected from randomly choosing tasks.

To demonstrate that task grounding with IBM2 successfully learned reasonable groundings of English words, we extracted IBM2's translation parameters after it had finished training on the original dataset. The most likely English words generated from the semantic word "robotInRoom" (which was associated with the lifted "go to room" task), were "walk," "through," "move," "go," and "from." "From" and "through" occurred because in "go to room" tasks, users often described *from* which room to leave and often commanded the robot to go *through* a door to the goal room. For example, one of the commands provided was "walk through doorway from orange room to beige room."

Since the color of rooms was typically used to describe the goal room of both lifted tasks, the words associated with it are especially relevant. The "roomIsOrange" semantic word was mostly likely to generate the words "red" and "orange;" "roomIsTan" was most likely to generate the words "tan" and "beige;" and "roomIsTeal" was most likely to generate "green" and "blue."

B. Expert Dataset

The expert dataset includes three kinds of goal-directed lifted tasks: one for going to a room; one for taking a block to a room; and one for taking a block to a room and then going to another room. Environments had either one or two blocks present, which could be either a chair or a bag, and three rooms whose colors varied between environments. The dataset consisted of many different verbal descriptions of tasks, which also resulted in different necessary object-binding constraints that would need to be inferred. For example, some expressions took the form of "move the bag to the blue room," which requires object-binding constraints for the shape of the block (bag) and the color of the destination room. Another example is "move bag to the room with chair," which requires inferring object-binding constraints for the shape of the target block, and a `blockInRoom` and a block-shape propositional function to disambiguate the target room. Different commands also had variable levels of specificity. For example, some included information that was not necessary to infer the correct task, such as "push chair from blue room to red room," in which specifying the color of the room in which the chair *initially* resides (blue) is unnecessary when there is only one chair. We also included commands that could not be modeled with the propositional functions defined. For example, the command "go to left room" is not representable because there are no propositional functions defined that indicate the relative spatial position of the rooms.

A strength of our framework is that different demonstrations of a command, even suboptimal ones, can be given and still provide meaningful information. In our dataset, we have 35 additional instances of a command given in the same environment, but with a different demonstration. However, to ensure that the LOO evaluation is always inferring behavior from either a novel command or a novel environment, we also created a version of the dataset that did not include any duplicate commands unless they were given in a different environment. We will refer to the dataset without multiple demonstrations in the same environment as *expert* and the dataset with multiple demonstrations as *expert-MD*. The expert dataset has 118 instances, with 104 unique commands (14 of the commands are the same as another, but are given in a different environment); expert-MD has 153 instances.

The LOO performance of the expert datasets is shown in Table II. As a baseline, if the agent randomly selected a possible grounded task, it would have an expected performance of 8.7% and 8.6% for the Expert and Expert-MD datasets, respectively. On both the Expert and Expert-MD dataset, our task model with IBM2 performed the best and these differences were statistically significant ($p = 0.013$ on a chi-squared test with all three groups). Furthermore, IBM2 performed better on Expert-MD than Expert (and the differences were statistically significant with $p = 0.011$), whereas BoW and Action Grounding did not improve performance by any significant amount, suggesting that even better performance with task grounding with IBM2 may be possible with additional data, even if that data uses different demonstrations than prior examples.

C. Transferring the Learned Model to a Different Robot

One of the strengths of grounding language to high-level tasks is that the learned model can be transferred to different robots with a different set of actions, because each robot can plan independently for each reward function. We demonstrate this transferability after training in our simulated version of the Cleanup World by taking the learned model and providing it to an actual mobile robot whose action space consists of three actions: turning 90 degrees clockwise or counterclockwise and

moving forward a fixed distance, rather than moving north, south, east, or west as in the simulator in which training was performed. To execute a policy, the robot needed to know its location and the location of block objects in the world, which we determined by using a motion tracking camera system. A video of the robot executing the AMT gathered command "bring star to beige room" can be found online[3] and images from the execution sequence are shown in Figure 5.

This demonstration highlights another advantage of our approach that grounds to tasks instead of actions: the real world introduces a number of sources of noise in the actuators and perception that make following an action sequence ineffective. In Figure 5b, we find that the robot has overshot, positioning itself directly behind the block. Even if we had successfully trained our model to ground to actions in this robot's action space, this failure in outcome expectation would cause the rest of the action sequence to fail to achieve the goal. Instead, because our model grounds commands to tasks and follows a corresponding policy, we find in Figure 5c that the robot has corrected itself and is then able to complete the task (Figure 5d).

VII. Related Work

Our work relates to the broad class of methods for grounded language learning that aim to ground words from a situated context. Instead of using annotated training data consisting of sentences and their corresponding semantic representations [13, 24, 27, 26], most of these approaches leverage non-linguistic information from a situated context as their primary source of supervision.

These approaches have been applied to various tasks, the ones closest to ours being interpreting verbal commands in the context of navigational instructions [23, 6, 10], robot manipulation [9, 22, 11], and puzzle solving and software control [3]. Reinforcement learning has been applied to train a policy to follow natural language instructions for software control and map navigation [23]. However, our goal is to move away from low-level instructions that correspond directly to actions in the environment to high-level task descriptions expressed using complex language. Unlike previous approaches that learn word groundings from pairs of natural language instructions and demonstrations of corresponding high-level actions [22, 9], our method learns mappings of natural language instructions to task descriptions represented as OO-MDP reward functions, enabling the agent to be autonomous and creative in its behavior. Misra et al. [18] show how to fill in the gaps when the learned actions do not directly match. Although this approach can improve generalization, our approach of mapping to reward functions enables the agent to infer creative solutions to problems that it encounters that may not have ever appeared in training. Our work is most closely related to Howard et al. [11]; both approaches map between language and an abstract, goal-based representations that enables creative solutions to problems. However, Howard et al. focus on mapping to planning constraints for motion planning, whereas our approach maps to reward functions that can represent different classes of problems and handle planning under uncertainty.

In addition, our generative model allows an investigation of multiple language models that can be used with the task model. Besides the generally used BoW model [2, 23], we showed that a word-based statistical machine-translation model provides better results. The idea of using statistical machine translation approaches for semantic parsing was introduced by Wong and Mooney [24] in a supervised learning setting, although they map to an action-based representation rather than using reward functions. In future work, we plan to move beyond word-based SMT models to grammar-based SMT models such as Synchronous Context-Free Grammars [25].

VIII. Conclusion

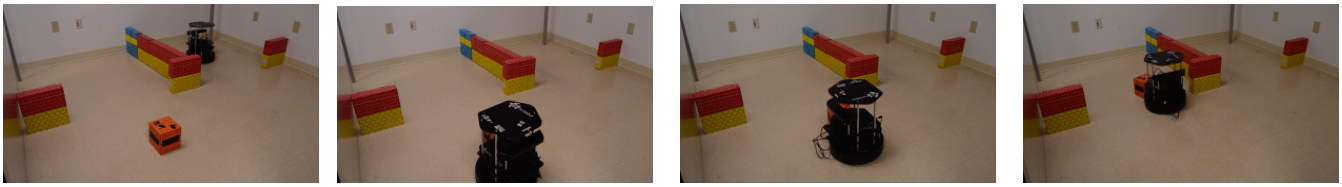
We presented a novel generative task model that expresses tasks as factored MDP reward functions, and to which natural language commands can be grounded. This generative task model is flexible and can be combined with different language models. We further showed how our model enables task groundings to be learned indirectly from example demonstrations by using IRL to weakly supervise the language model learning. Grounding natural language commands to MDP tasks rather than actions is a powerful approach because it allows people to provide robots high-level commands without specifying the details of how to complete the tasks. This task grounding enables the robot to correctly follow a command in a variety of environments, including environments it has not previously encountered, and adapt to noise in the environment, as shown in our results. In contrast, grounding language to action sequences fails to generalize to novel environments. Moreover, grounding language to tasks allows the learned model to be transferred to different robots that have different actions spaces than the robot used to train the model, which was demonstrated by transferring the learned language model onto a physical robot with a different action space than the action space of the simulated robot used in training.

The grammar-free models we used here have the advantage of not requiring a corpus for training the grammar model, and they were sufficient for successfully grounding the language to tasks in our test domain. However, there may be other domains in which more complex grammar-based language models could provide better results.

In this work, our robotics task was fairly simple to facilitate gathering commands from an AMT study. However, because our approach is independent of the planning algorithm learning, and because only the task description (rather than the state space) needs to be symbolic, we believe our approach can scale to a variety of more complex robotics tasks, as long as enough of the state is observable to perform task grounding. For example, we believe that this work will scale to robotics domains like forklift manipulation [20].

Currently, our tasks are described as a logical expression over a set of free variables. In the future, this could be

[3]https://vid.me/Wfxx



(a) (b) (c) (d)

Fig. 5: Images from our mobile robot executing the command "bring star to beige room." The room colors are hard coded to the different regions outlined by the walls (top left, right, and bottom room colors are beige, orange, and teal, respectively) and the orange block is mapped to the "star" object that AMT users were shown. The initial state is shown in image (a). Image (b) shows that the robot overshot positioning itself behind the block due to noise in the real world. In image (c), the robot has corrected its position and it completes the task successfully in image (d) after pushing the block into the correct room.

extended to include first-order logic quantifiers. Because tasks are represented with reward functions, our approach could also incorporate continuing tasks, rather than goal-directed tasks.

Task grounding requires the agent to know about the state of the objects a human refers to in their command. In the future, it would be useful to relax this constraint so that the robot can infer the existence of unobserved objects in the world being described. This functionality might be incorporated by adding to the generative model the generation of additional objects not observed in the input state.

Finally, although our ultimate goal is for the robot to fully solve the tasks itself to remove the burden from the human, in particularly challenging tasks, it may useful to allow the human to specify subgoals that simplify the planning task for the robot.

IX. Acknowledgments

This work was supported by collaborative NSF awards #1065195 and #1340150. We would like to thank Eugene Charniak, who suggested IBM Model 2 for this work.

References

[1] Monica Babeş-Vroman, Vukosi Marivate, Kaushik Subramanian, and Michael Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the Twenty Eighth International Conference on Machine Learning (ICML 2011)*, 2011.

[2] S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, 2009.

[3] S.R.K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL 2010)*, 2010.

[4] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16 (2):79–85, June 1990. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=92858.92860.

[5] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972474.

[6] David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011).*, pages 859–865, 2011.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[8] Carlos Diuk, Andre Cohen, and Michael Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML-08)*, 2008.

[9] Felix Duvallet, Thomas Kollar, and Anthony (Tony) Stentz. Imitation learning for natural language direction following through unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[10] Alexander Grubb, Felix Duvallet, Stephanie Tellex, Thomas Kollar, Nicholas Roy, Anthony Stentz, and J. Andrew Bagnel. Imitation learning for natural language direction following. In *Proceedings of the ICML Workshop on New Developments in Imitation Learning*, 2011.

[11] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2014.

[12] Andreas Junghanns and Jonathan Schaeffer. Sokoban: A challenging single-agent search problem. In *In IJCAI Workshop on Using Games as an Experimental Testbed for AI Reasearch*. Citeseer, 1997.

[13] Rohit J. Kate and Raymond J. Mooney. Using string-kernels for learning semantic parsers. In *Proceedings of*

the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, ACL-44, 2006.

[14] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proceedings of HRI-2010*, 2010.

[15] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Proceedings of the 13th International Symposium on Experimental Robotics (ISER)*, 2012.

[16] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the 2012 International Conference on Machine Learning*, Edinburgh, Scotland, June 2012.

[17] Andrew McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI'99 Workshop on Text Learning*, pages 1–7, 1999.

[18] DK Misra, Jaeyong Sung, Kevin Lee, Ashutosh Saxena, Jaeyong Sung, Bart Selman, Ashutosh Saxena, Jaeyong Sung, Colin Ponce, Bart Selman, et al. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems, RSS*, 2014.

[19] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[20] Seth Teller, Matthew R Walter, Matthew Antone, Andrew Correa, Randall Davis, Luke Fletcher, Emilio Frazzoli, Jim Glass, Jonathan P How, Albert S Huang, et al. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 526–533. IEEE, 2010.

[21] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificfical Intelligence*, 2011.

[22] Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):205–232, 2014.

[23] Adam Vogel and Dan Jurafsky. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL 2010)*, 2010.

[24] Yuk Wah Wong and Raymond Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-2007)*, 2007.

[25] Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

[26] Luke Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Association for Computational Linguistics (ACL'09)*, 2009.

[27] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI-05*, 2005.