Learning of Complex-Structured Tasks from Verbal Instruction

Monica Nicolescu[1], Natalie Arnold[2], Janelle Blankenburg[3], David Feil-Seifer[4],
Santosh Banisetty[5], Mircea Nicolescu[6], Andrew Palmer[7], Thor Monteverde[8]

Abstract— **This paper presents a novel approach to robot task learning from language-based instructions, which focuses on increasing the complexity of task representations that can be taught through verbal instruction. The major proposed contribution is the development of a framework for *directly mapping a complex verbal instruction to an executable task representation*, from a single training experience. The method can handle the following types of complexities: 1) instructions that use conjunctions to *convey complex execution constraints* (such as alternative paths of execution, sequential or non-ordering constraints, as well as hierarchical representations) and 2) instructions that use prepositions and multiple adjectives to *specify action/object parameters relevant for the task*. Specific algorithms have been developed for handling *conjunctions*, *adjectives* and *prepositions* as well as for translating the parsed instructions into parameterized executable task representations. The paper describes validation experiments with a PR2 humanoid robot learning new tasks from verbal instruction, as well as an additional range of utterances that can be parsed into executable controllers by the proposed system.**

I. Introduction

This paper presents a novel approach for robot task learning from human instruction, which focuses on increasing the complexity of the task representations that can be learned. In particular, we consider two main types of instruction complexities: 1) instructions that use conjunctions to *convey complex execution constraints* (such as alternative paths of execution, sequential or non-ordering constraints, as well as hierarchical representations) and 2) instructions that use prepositions and multiple adjectives to *specify action/object parameters relevant for the task.*

Existing research on teaching robots by demonstration or verbal instruction focuses on learning tasks that mainly involve sequential constraints, building representations that encode steps which have to be executed in order. In practice, robot tasks may require more complex dependencies. For instance, some parts of the task could be allowed to execute in any order (e.g., adding ingredients for making cookies), leading to multiple ways in which the task can be performed. Other parts of the task may have to be executed in a specific order (e.g., adding ingredients before doing the mixing). Furthermore, other parts of the task could be achieved through entirely different paths of execution (e.g., could add either whole wheat, or white flour, or almond flour in a recipe). Such tasks are difficult for a human to teach by demonstration [1], as in order to capture the various different ways in which a task may be executed, a learning system may need to be provided with multiple demonstrations of the same task. In contrast, these types of complex dependencies can be efficiently conveyed by use of conjunctions in verbal instructions in a single command. This work proposes algorithms that process such instructions and produce a hierarchical task representation that encapsulates the execution constraints and is directly executable by the robot.

To take advantage of the richness provided by natural instruction, a key requirement is that the learned tasks be parameterized to the full extent possible. Features such as desired target location for a specific object movement or placement, as well as attributes of one or more objects involved are essential for proper specification of collaborative manipulation tasks. This type of information is provided through conjunctions *and, or*, prepositions such as *on, behind, below, underneath*, as well as adjectives such as *small, red, tall*, which give specificity regarding a target object for the task. This paper presents algorithms that parse the instruction and automatically link the prepositions with the parameters of existing behaviors and attaches the adjectives to the referred objects in the generated task controller.

Given the focus on teaching robots new tasks using linguistic instructions, this work is solely focused on parsing imperative sentences. The main contribution is the ability to automatically convert such instructions, with the complexities described above, into a robot task controller that can be directly used for execution.

II. Related Work

Numerous approaches have been designed for translating natural language instructions into control structures, focusing on various aspects of grounding linguistic information onto physical actions, objects or other relevant attributes. In this work we focus on the specific problems of 1) learning representations that encode complex execution constraints (provided through conjunctions) and of 2) parameterizing learned tasks from information provided by adjectives and prepositions.

Despite the rich spectrum of instruction-based task learning approaches, existing methods encapsulate mostly sequential constraints, as a series of individual steps that have to be performed in a particular given order. [2] shows a first example of controlling a robot using instructions given in natural language. The system uses an explicitly defined grammar that is domain dependent, restricting the robot to understanding

Department of Computer Science & Engineering University of Nevada, Reno, Reno, NV, 89557, USA

[1]Email: monica@cse.unr.edu, [2]Email: narnold@nevada.unr.edu, [3]Email: jjblankenburg@nevada.unr.edu, [4]Email: dave@cse.unr.edu, [5]Email: santoshbanisetty@nevada.unr.edu, [6]Email: mircea@cse.unr.edu, [7]Email: swordsofold@gmail.com, [8]Email: twickemonteverde@nevada.unr.edu

only instructions related to navigation. Similarly, [3] and [4] present systems that focus on guiding a robot through natural language to navigate in the environment, focusing on issues related to spatial representation of the world and navigation actions. Other approaches have focused on problems such as pick-and-place [5], grasping [6], blocks worlds [7], building of motion controllers [8], or navigation, delivery and validation tasks [9]. Chang [10] presents an approach to increase the flexibility of a speech-based interface, by having the system learn to associate complex desired configurations with particular simple instructions (such as what lights need to be turned on and to what level for "reading mode" or "TV mode"). However, the configurations are specified by the user and are next associated with a simpler command that is used to identify that situation. Arumugam et al. [11] present a method for grounding verbal instructions at varying degrees of specificity using a deep neural network language model that selects the appropriate level of a planning hierarchy. Methods focused on reinforcement learning and neural networks in simulated domains have also been developed to combine visual input and instructions to actions [12], [13], to ground language commands to reward function represented by a deep neural network [14], or to provide iterative language corrections [15]. A related approach for one shot learning of actions and new objects from language instruction has been proposed in [16], in which new tasks are learned as sequences of individual actions. In this work we aim to use the flexibility of natural language that can concisely include multiple dependencies in a single command (for example, *"Do a THEN b OR c OR d"*) to enable the learning of more complex task representations.

Several approaches have been developed with a focus on increasing the generality of the tasks learned: MacGlashan et al. [17] learns mappings of natural language instructions to task descriptions encoded as Object-Oriented Markov Decision Processes that generalize to new environments and to robots with new action spaces. Howard et al. [18] create mappings to planning constraints that are used to generate a sequence of action that represents the instruction. The tasks learned through our method inherently generalize to different environments and in particular enable opportunistic ways of task execution based on the specifics of the environment.

Recent work closely related to our goals is presented in [19]. The approach relies on a library of verb-environment-instructions built from a data set of task descriptions, which represents all possible instructions for each verb in that environment. Relying on this, a model dependent on an energy function resolves ambiguities based on appropriate environment context and task constraints. A framework on Generalized Grounding Graphs (G3) is presented in [20], for both navigation and object manipulation. The framework allows for dynamic instantiation of a probabilistic graphical model for a given natural language command, taking into account the hierarchical and compositional semantic structure of the instruction. The method relies on a corpus of sentences specific to the manipulation task to infer the most likely meaning of the instruction. We propose a method that does not require training or a corpus specific to a particular task: we use the argument information provided by a semantic parser [21] to automatically generate task controllers for an unrestricted set of action verbs.

Other relevant work is presented in [22], which presents a method for interactive task learning that combines natural language communication and action demonstration to teach physical agents new tasks. The approach focuses on commonsense knowledge (physical causality knowledge) in order to enable the grounding of language to perception and action. The learned tasks have a similar hierarchical structure that encodes mainly sequential task execution. Our focus is on learning tasks with multiple different types of execution constraints, on being able to extract knowledge for parameterizing behaviors from prepositions and attributes, and demonstrating an integrated system that is validated on a physical robot.

An approach that aims to handle reference expressions and prepositions is described in [23]. Object manipulation actions are represented with a pre-defined set of commands and prepositions, and fixed offsets representing object placement positions. Similar to our goals, [24] describes an approach for learning executable robot plans from online instructions of manipulation tasks. Instructions that contain prepositions, such as *"Place the cup on the table"* can also be handled. Although instructions are mapped to executable representations, the learned tasks are not validated on physical or simulated robots. The use of attributes (such as color) for ambiguity resolution and identifying objects in the physical environment is shown in [25]. The method is an extensive integrated approach for grounding interactions, but does not focus on robot task learning from language.

Different from the above methods, our approach aims to parse general verbal instructions that use attributes and prepositions and concisely encode multiple execution dependencies. Furthermore, the focus of this work is not on handling the full spectrum of unstructured linguistic instructions and ambiguities, but rather on handling the more restricted domain of imperative sentences that is typical for teaching by instruction: for this, the teacher aims to convey the information to the learner as clearly as possible, aiming to minimize ambiguity in order to facilitate the learning process.

III. Learning from Verbal Instructions

A. Hierarchical Task Representation

We previously developed a robot control architecture [26] that brings the following main contributions: 1) it provides an efficient, compact encoding of tasks with multiple execution constraints, 2) it uses the same compact representation as the controller that the robot will use to achieve its goals, 3) it allows the robots to dynamically decide which execution path to follow using an activation spreading mechanism that relies on environmental conditions, and 4) it provides robustness to changes in the environment during the task execution.

The representation is built using a behavior-based paradigm [27] and enables the system to encode tasks

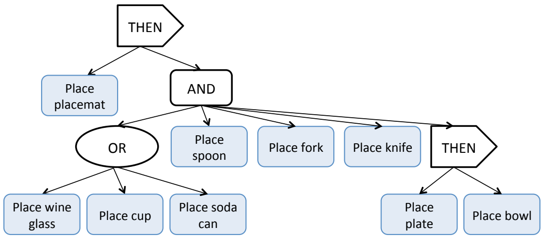
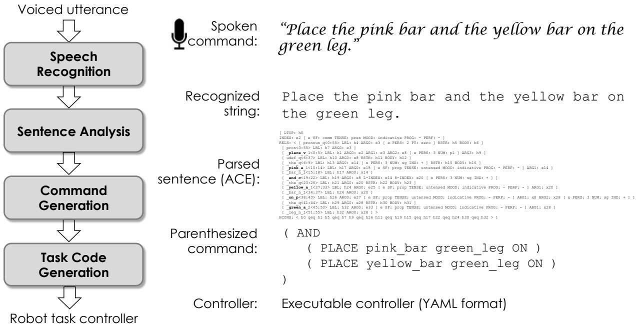


Fig. 1. Task representation for setting up a table.

involving various types of constraints such as *sequential*, *non-ordering*, and *alternative paths* of execution. All of these constraints can be incorporated into a single task representation such as that presented in Figure 1. To encode such a task we define two types of nodes in our behavior network: **goal nodes** and **behavior nodes**. **Goal nodes** provide the base goal control behaviors of the hierarchical task structure, and include the *THEN*, *AND*, and *OR* nodes that are used internally in the tree to encode the task constraints: i) **THEN** is an n-ary node used to encode sequential constraints (each child must execute before the children to its right can execute), ii) **AND** is an n-ary node used to encode non-ordering constraints (children can be executed in any order), and iii) **OR** is an n-ary node used to encode alternative paths of execution (only one of the children will be executed). **Behavior nodes** are the leaf nodes in the task tree structure and encode the physical behaviors that the robot can perform, e.g. a *Place(Cup)* behavior will control the arm of the robot to pick up a cup from the table in front of it and place it in another location. While in [26] the task representations were manually designed, in this paper we present an approach for *automatically learning such task representations from verbal instructions*.

B. Learning of Task Controllers from Verbal Instruction

We assume that the robot is equipped with a set of basic skills (or behaviors), each of which has a mapping to the teacher's instruction vocabulary. In addition, the robot has knowledge of various objects (and their attributes) that can be appropriately recognized and manipulated. The teacher's instruction is parsed and mapped into an executable controller as outlined in Figure 2:



"Place the pink bar and the yellow bar on the green leg."

```
Place the pink bar and the yellow bar on
the green leg.
```

```
[ LTOP: h0
INDEX: e2 [ e SF: comm TENSE: pres MOOD: indicative PROG: - PERF: - ]
RELS: < [ pronoun_q<0:55> LBL: h4 ARG0: x3 [ x PERS: 2 PT: zero ] RSTR: h5 BODY: h6 ]
 [ pron<0:55> LBL: h7 ARG0: x3 ]
 [ _place_v_1<0:5> LBL: h1 ARG0: e2 ARG1: x3 ARG2: x8 [ x PERS: 3 NUM: pl ] ARG3: h9 ]
 [ udef_q<6:37> LBL: h10 ARG0: x8 RSTR: h11 BODY: h12 ]
 [ _the_q<6:9> LBL: h13 ARG0: x14 [ x PERS: 3 NUM: sg IND: + ] RSTR: h15 BODY: h16 ]
 [ _pink_a_1<10:14> LBL: h17 ARG0: e18 [ e SF: prop TENSE: untensed MOOD: indicative PROG: - PERF: - ] ARG1: x14 ]
 [ _bar_n_1<15:18> LBL: h17 ARG0: x14 ]
 [ _and_c<19:22> LBL: h19 ARG0: x8 L-INDEX: x14 R-INDEX: x20 [ x PERS: 3 NUM: sg IND: + ] ]
 [ _the_q<23:26> LBL: h21 ARG0: x20 RSTR: h22 BODY: h23 ]
 [ _yellow_a_1<27:33> LBL: h24 ARG0: e25 [ e SF: prop TENSE: untensed MOOD: indicative PROG: - PERF: - ] ARG1: x20 ]
 [ _bar_n_1<34:37> LBL: h24 ARG0: x20 ]
 [ _on_p<38:40> LBL: h26 ARG0: e27 [ e SF: prop TENSE: untensed MOOD: indicative PROG: - PERF: - ] ARG1: x8 ARG2: x28 [ x PERS: 3 NUM: sg IND: + ] ]
 [ _the_q<41:44> LBL: h29 ARG0: x28 RSTR: h30 BODY: h31 ]
 [ _green_a_2<45:50> LBL: h32 ARG0: e33 [ e SF: prop TENSE: untensed MOOD: indicative PROG: - PERF: - ] ARG1: x28 ]
 [ _leg_n_1<51:55> LBL: h32 ARG0: x28 ] >
HCONS: < h0 qeq h1 h5 qeq h7 h9 qeq h26 h11 qeq h19 h15 qeq h17 h22 qeq h24 h30 qeq h32 > ]
```

```
( AND
    ( PLACE pink_bar green_leg ON )
    ( PLACE yellow_bar green_leg ON )
)
```

Executable controller (YAML format)

Fig. 2. Stages of parsing verbal instructions to controllers.

The *speech recognition* module takes as input a voice command from the user through the PocketSphinx package in ROS and produces a string representing the user's command. The *sentence analysis* module takes the command string and produces a parsed representation of the command. This is then used by the *command generation* module, which produces a parenthesized version of the command. In turn, this is next used by the *task code generation* module that produces the executable controller, in the form of a YAML file. These modules are described in more detail below.

For *sentence analysis*, in order to represent the semantic roles of each used utterance, we use *minimal recursion semantics* (MRS), which are based on the English Resource Grammar open source project [28], [29]. In MRS the links between meaningful words are shown through argument roles and handle links, which can capture some scope ambiguity. To extract the MRS representations of the verbal command, we used the Answer Constraint Engine (ACE) tool, along with its pre-compiled grammar, available at [21]. The engine also tags each word with part-of-speech information, which will be used in the next step of our processing and will be described in detail below.

The *command generation* module takes as input the parsed sentence representation, and produces a parenthesized form of the command (Figure 2) as follows. The semantic representation produced by ACE is parsed to extract relational information for each of the words in the sentence, which is then organized in a dictionary of relations (`RELS`) with the following structure: `Handle: [category, word, [arguments]]`, as shown in Figure 3. The *Handle* is a unique identifier given to the word by the ACE analyzer consisting of a letter and a number (the `ARG0` of each relation, for example, `x10` corresponds to the noun `bar`). The *category* represents the part of speech of the word (e.g., noun, verb, conjunction, etc.) and the *arguments* is a list of relations (`ARG0-ARGn`) to other words in the sentence. Each part of speech has a different number of arguments, as follows. Conjunctions have two arguments, each pointing to the items that they connect. For example, the conjunction `'and'` has arguments `x10, x16` (representing the first, and respectively the second noun `'bar'` it connects). Verbs have three arguments, but only the second one (`ARG2`) is relevant for our purpose: this argument points to the handle of either a noun or a preposition that links several nouns. For example, the verb `'place'` has `ARG2 = x4`, which is the conjunction `'and'` that links two nouns. Prepositions have two arguments, but only the second one (`ARG2`) is relevant, indicating the object of the preposition. For instance, the preposition `'on'` has `ARG2 = x24`, which is the noun `'leg'`. Adjectives have only one argument `ARG1`, which indicates the object they refer to (e.g., adjective `e21`, representing `'yellow'`, refers to relation `x16`, which is the noun `'bar'`). Nouns do not have any arguments, except for their `ARG0` name.

Algorithm 1, *MRS_Crawling(sentence, RELS)* takes as input the parsed sentence produced by ACE and the `RELS` dictionary and starts by finding the sentence index (`e2` in our case), and the semantic relation to which it corresponds. In our sentence this is represented by the verb `'place'`, which should correspond to one of the robot's basic behaviors (lines

```
{'e2': ['verb', 'place', ['i3', 'x4', 'h5']],
'e14': ['adjective', 'pink', ['x10']],
'x10': ['noun', 'bar', []],
'x4':  ['conj', 'and', ['x10', 'x16']],
'e21': ['adjective', 'yellow', ['x16']],
'x16': ['noun', 'bar', []],
'e23': ['preposition', 'on', ['x4', 'x24']],
'e29': ['adjective', 'green', ['x24']],
'x24': ['noun', 'leg', []]}
```

Fig. 3. Dictionary of relations (RELS) extracted for command generation.

Algorithm 1 $MRS\_Crawling(sentence, RELS)$

```
1: $index\ = sentence.INDEX\ \ //chose\ sent.\ index$
2: $look\ for\ semantic\ relation\ (in\ RELS)$
   $with\ rel.ARG0\ ==\ index$
3: verb = $rel\ \ //(this\ is\ the\ action\ verb)$
4: $ConnectAdjectives(RELS)$
5: $HandlePrepositions(RELS)$
6: $command\ =\ BuildCommand(RELS, verb)$
```

1-3). Next, adjectives are appended to their corresponding nouns, indicated by their `REL1` argument, as shown in Algorithm 2, *ConnectAdjectives(RELS)*. After this processing, the `word` fields for the nouns in the dictionary become `′pink_bar′`, `′yellow_bar′`, and `′green_bar′`.

Algorithm 3, *HandlePrepositions(RELS)* processes all the prepositions in the dictionary to build relations of the type `<subject object preposition>`. The object of the preposition is obtained from the preposition's `ARG2`. The subject of the preposition is found in `ARG0`, and can be either a single noun or a conjunction (as in the example: both the pink bar and yellow bar are the subjects placed on the green bar). Conjunction objects are recursively found by Algorithm 4, *FindAllSubjects(rel)*, which takes as input one of the relations in the RELS dictionary. After this processing stage, the `word` field for the preposition's subject nouns in the dictionary become `′pink_bar green_leg on′`, `′yellow_bar green_leg on′`.

The *controller construction* module takes as input the parenthesized form of the task representation and translates it into a robot controller that can automatically be executed by the robot, using the procedure shown in Algorithm 6. The input to the algorithm is a fully parenthesized string (Figure 2) and the output is a node with its corresponding list of children. The *GetCrtToken()* function just reads from the string the next relevant element (either a parenthesis, a node label such as *THEN, OR*, or a parameter such as *cup, tea*). The *AdvanceToNextToken()* advances to the next token in the string command. The algorithm proceeds with extracting the opening parenthesis, then the node label and initializes the list of children to be empty (lines 1-6). After this, the algorithm repeatedly processes the next tokens until reaching a closing parenthesis *′)′*. The new tokens could be either new nodes themselves (lines 8-12), or behavior parameters (lines 8, 15-18). When the token is a closing parenthesis *′)′*, there is nothing to be done and the loop stops (lines 13-14, 21). The newly created node with the list of its children is returned (line 22).

Algorithm 2 $ConnectAdjectives(RELS)$

```
1: for all rels in RELS do
2:   if $rel.category\ ==\ adjective$ then
3:     //get handle for noun
4:     $noun\_handle\ = rel.ARG1$
5:     //append adjective to noun
6:     $noun\_handle.word += ``\_" + rel.word$
7:   end if
8: end for
```

Algorithm 3 $HandlePrepositions(RELS)$

```
1: for all rels in RELS do
2:   if $rel.category\ ==\ preposition$ then
3:     $main_{s}ubject\ =\ rel.ARG1$ //get source of preposition
4:     $subjects\ =\ FindAllSubjects(main\_subject)$
5:     $object\ =\ rel.ARG2$ //get object of preposition
6:     for all sbj in subjects do
7:       //append preposition and subject noun
8:       $sbj.word\ +=\ ``" + object.word + ``" + rel.word$
9:     end for
10:  end if
11: end for
```

Algorithm 4 $FindAllSubjects(rel)$

```
1: for all arg in rel.ARG0 do
2:   if $rel.category\ ==\ noun$ then
3:     $Appendreltosubjects$ //append source noun
4:   else if $rel.category\ ==\ conjunction$ then
5:     //recursively find conjunction-connected subjects
6:     subjects_list = FindAllSubjects(rel.arg)
7:     $Append\ subjects\_list\ to\ subjects$
8:   end if
9: end for
```

C. Learning of Basic and High-Level Tasks

While single instructions can have a high-degree of complexity (as shown in Section III-B), they may only represent a part of a larger task to be learned. We developed an approach to allow the robot to learn tasks composed of such multiple instructions, as well as to build up from those tasks in order to learn even more complex representations. In this work we differentiate between *basic* and *high-level* tasks. *Basic tasks* are built entirely from combinations of low-level skills (behaviors) of the robot. *High-level* tasks are built from combinations of already existing basic tasks, or other high-level tasks previously learned.

The process for learning *basic tasks* consists of three main steps that run in a loop until the teacher is done with teaching the task. In each iteration of the loop, the robot receives a verbal instruction from the teacher, which is next processed and converted into an executable controller, as described in Section III-B. An instruction can be as simple as a basic

Algorithm 5 $BuildCommand(RELS, handle)$

```
$command\ =\ empty$
$verb = handle.word$
$look\ for\ semantic\ relation\ (RELS)$
    $with\ rel.ARG0 == verb\_handle.ARG2$
if $rel.category\ ==\ noun$ then
    $command\ =\ “(”\ +\ verb\ +\ rel.word\ +\ “)”$
    return $command$
else if $rel.category\ ==\ conjunction$ then
    $current\_conj = rel.word$
    //open parenthesis, then append conjunction
    $command\ =\ “(”\ +\ current\_conj$
    //append left and right arguments for conjunction
    $command\ +=\ BuildCommand(verb, RELS, rel.ARG1)$
    $command\ +=\ BuildCommand(verb, RELS, rel.ARG2)$
    $command += “)”$ //append closing parenthesis
end if
return $command$
```

Algorithm 6 CreateNode(cmd)

```
$new\ node$ // create new node object
$token = GetCrtToken(cmd)$ // must be ’(’
$AdvanceToNextToken(cmd)$
$node.Label = GetNextToken(cmd)$
$AdvanceToNextToken(cmd)$
$node.ChildrenList = emtpy$
repeat
    $token = GetCrtToken(cmd)$
    if token == ’(’ then
        // child is a new node
        $child = CreateNode(cmd)$
        $node.AddToChildren(child)$
    else if token == ’)’ then
        // the end, do nothing
    else
        // child is a parameter
        $child = token$
        $node.AddToChildren(child)$
    end if
    $AdvanceToNextToken(cmd)$
until token == ’)’
$return\ node$
```

command (e.g., *Place the bread*), or could have a higher degree of complexity (such as *Place the cup, then the sugar and the tea*). Third, the newly learned step is executed by the robot before the teacher provides the next instruction. When the teacher finishes the training, all the individual steps (if more than one) are combined into a single task representation, which consists of a *THEN* root node, whose children are the nodes representing each individual step of the task in the order in which they have been presented. To facilitate a flexible and natural interaction during learning, both the human and the robot use specific verbal cues to indicate the following: 1) by the human: when a training task begins, the name of the task, the end of the task, 2) by the robot: confirmation of proper instruction received, request of names for newly trained tasks, requests for new steps, or request to repeat the task if the command is not properly understood by the speech recognition module. Examples of full dialog sequences between the human and the robot during training are presented in Section IV.

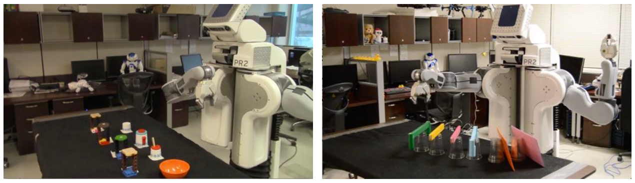


Fig. 4. Experimental setup. Left: household, Right: Ikea Eket base.

To learn *high-level* tasks, the teacher provides instructions that combine tasks already existing in the robot’s repertoire. The process runs in a loop in which verbal instructions are provided, then processed and converted into an executable controller, similar to the process for basic tasks. If the instruction includes reference to a task previously learned, the robot does not execute the individual command after it is received, but rather waits for the training to finish. This is an arbitrary choice we made to distinguish the two cases, but has no influence on the learning process. At the end of the training, to build the task representation for the entire task, all the individual steps are combined into a single task representation as in the case for the basic tasks.

IV. Experimental Validation

A. Robot Experiments

We validated our approach with a PR2 humanoid robot in two scenarios: a household environment (in which we validate the learning of basic and high-level tasks) and an Ikea Eket base frame construction environment (in which we validate the use of prepositions and adjectives to parameterize the robot’s behaviors).

1) Household Environment: In the household environment, the robot was taught to perform the following tasks: 1) two basic tasks, one for making tea, and one for making a sandwich and 2) a high-level task called tea-time that consists of the two basic tasks for making tea and a sandwich, using toy objects that represent *bread*, *meat*, *lettuce*, *tea*, *cup*, and *sugar*. The robot is equipped with a *Place(Object)* behavior, that is parameterized for the different objects. Since the focus of this work is on the learning approach, for these experiments the initial and final locations of the objects are pre-defined. The experimental setup is shown in Figure 4 (left).

The first two training experiments were focused on teaching the two basic tasks. Table I shows the verbal communication between the human and the robot during the training experiment, as well as the robot’s actions during this process. At the end of this training, the robot has a new task in its repertoire, called **sandwich** (left branch of the root *AND* node in Figure 5), which can be executed at any time or used as a part of a higher-level task. During the practice of the command *“Place the meat or the lettuce”*, the robot chose to select the *meat* out of the two possible options, due to the fact that it was closer to the robot hand than the *lettuce*.

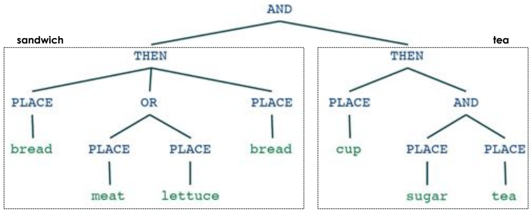


Fig. 5. Hierarchical representation for the **tea-time** task.

Step	Dialog and Actions
1.	**H:** *“Make recipe.”*
2.	**R:** *“Ok, make recipe. What is the name of the recipe?”*
3.	**H:** *Make **sandwich.***
4.	**R:** *“Ok, recipe for **sandwich**. Please start with the instructions.”*
5.	**H:** *“Place the left bread.”*
6.	**R:** *“Ok, place the left bread.”* Builds controller and executes the task: (PLACE Bread) **R:** *“What’s the next step?”*
7.	**H:** *“Place the meat or the lettuce.”*
8.	**R:** *“Ok, place the meat or the lettuce.”* Builds controller and executes the task: (OR (PLACE Meat) (PLACE lettuce)) **R:** *“What’s the next step?”*
9.	**H:** *“Place the right bread.”*
10.	**R:** *“Ok, place the right bread.”* Builds controller and executes the task: (PLACE Bread) **R:** *“What’s the next step?”*
11.	**H:** *“Store recipe.”*
12.	**R:** *“Ok, recipe for **sandwich** is complete and stored.”* Final task representation stored (Fig. 5, left *THEN* branch).

TABLE I
HUMAN-ROBOT DIALOG AND INTERACTION DURING TRAINING OF THE **SANDWICH** TASK.

Using a similar process, in the second experiment the robot is trained a task for making **tea**, the representation of which is shown in Figure 5 (right branch). During the practice of this task, the ordering constraints have been enforced, with the robot placing the *cup* first and then proceeding to the next two objects. Since placing the *sugar* and *tea* do not have any ordering constraints, the robot chooses a path of execution based on the state of the environment, placing the *tea* first as it was closer.

The third experiment consists of teaching the robot a higher-level task, which combines tasks that are already known to the robot. Table II shows the verbal communication between the human and the robot during training. The task representation that is learned by the robot is shown in Figure 5. Since the teacher has provided a single instruction that combines two existing tasks by an *AND* conjuction, the full task representation consists of a single *AND* node as the root of the task tree.

Step	Dialog and Actions
1.	**H:** *“Make recipe.”*
2.	**R:** *“Ok, make recipe. What is the name of the recipe?”*
3.	**H:** ***Tea time.***
4.	**R:** *“Ok, recipe for **tea time**. Please start with the instructions.”*
5.	**H: Sandwich** and **tea**
6.	**R:** *“Ok, **sandwich** and **tea**. What else?”* Builds controller for **sandwich** and **tea** (Figure 5).
7.	**H:** *“Store recipe.”*
8.	**R:** *“Ok, recipe for **tea time** is complete and stored.”* Executes the task.

TABLE II
HUMAN-ROBOT DIALOG AND INTERACTION DURING TRAINING OF THE **TEA-TIME** TASK.

2) IKEA Eket Base Assembly: In the previous scenario, the robot’s *Place(Object)* behavior was limited to a single parameter, which was the object of place action. By enabling the parsing of prepositions and adjectives, the behaviors can be parameterized with destination location (e.g., *on the green leg*) as well as specifics of the objects involved (e.g., *pink leg*). The task consists of building an Ikea Eket base that has two bars that need to be placed on two legs. One of two tops (purple and orange) can be selected to be put on top of the built base. For easy recognition by a vision system and to showcase the use of adjectives, the legs were painted with green and blue, and the bars were painted with yellow and pink. Table III shows the human instructions (without the robot’s responses, which follow the same pattern as in the previous examples).

Step	Human Instructions and Robot Actions
1.	**H:** *“Place the green leg in front of you.”*
2.	**H:** *“Place the pink bar and the yellow bar on the green leg.”*
3.	**H:** *“Place the blue leg onto the base.”*
4.	**H:** *“Place the orange top or the purple top on the base.”*

TABLE III
HUMAN-PROVIDED INSTRUCTIONS DURING TRAINING OF THE **EKET** TASK.

Figure 6 shows the hierarchical representation of the learned task. During task execution, the location information provided by the prepositions (e.g., *FRONT-OF, ON, ONTO*) is mapped to specific positions of the source object with respect to the destination object, based on a pre-determined table. Our current vision system does not yet provide pose information for the objects and the offsets of the prepositions give rough placement positions. Therefore, at task execution the robot asks a human user for assistance to precisely position the objects with respect to each other before making the final assembly. In future work this will be addressed by integrating a vision-based system that provides pose information as well as by specifying locations in a finer grain of detail, for example that a particular side of a bar should fit on a particular side of the leg.

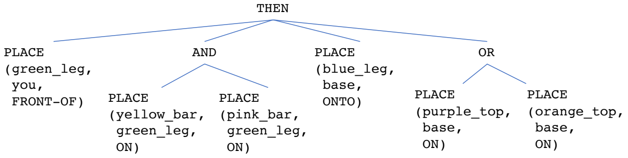

THEN
PLACE (green_leg, you, FRONT-OF)
AND
PLACE (blue_leg, base, ONTO)
OR
PLACE (yellow_bar, green_leg, ON)
PLACE (pink_bar, green_leg, ON)
PLACE (purple_top, base, ON)
PLACE (orange_top, base, ON)

Fig. 6. Representation of the learned Eket assembly task.

B. General-Purpose Task Learning Experiments

This section provides additional results that demonstrate in more detail the full capabilities of the approach for parsing language instructions to controllers. These tasks have not been validated on a robotic system, but show the representational power of the approach.

1) Complex Task Execution Constraints: This section presents additional examples of complex task representations that can be constructed from a single verbal instruction. Figures 7- 9 show the task tree as well as the parenthesized command for several commands.

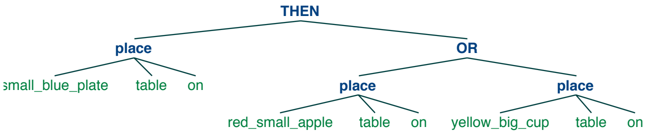


Fig. 7. Instruction: "Place the blue small plate then the small red apple or the big yellow cup on the table."

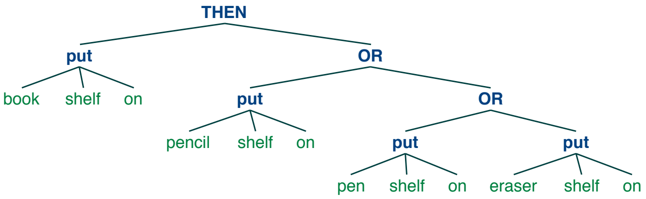


Fig. 8. Instruction: "Put the books then the pencil or the pen or the eraser on the shelf."

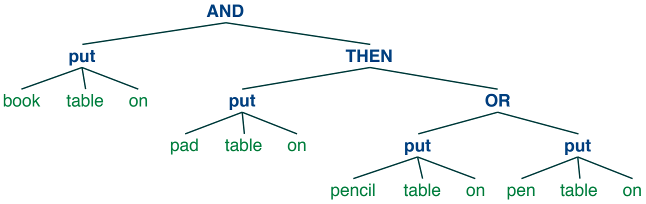


Fig. 9. Instruction: "Put the book and the pad then the pencil or the pen on the table."

All these examples include temporal sequencing constraints (shown by the *THEN* nodes), non-ordering constraints (shown by the *AND* nodes), as well as alternative paths of execution (shown by the *OR* nodes). In order to learn these constraints using demonstrations/instructions that solely rely on sequential commands, the system would need to be provided with multiple demonstrations that illustrate all the alternative ways of execution and sequencing/non-sequencing constraints. The proposed method of mapping verbal instructions into controllers that encode the constraints provided by the *THEN, AND, OR* conjunctions, enables complex execution constraints to be conveyed to a robot in a single instruction.

2) Use of Adjectives: The proposed system can handle all descriptive adjectives, and multiple adjectives can be used to refer to the same noun. Positive, comparative and superlative are also successfully parsed. In contrast, the following adjectives are not currently handled: quantitative adjectives (*some, few, all*), demonstrative adjectives (*this, that, these*), possessive adjectives (*my, your, his*), distribute adjectives (*each, every, either*). Handling these types of adjectives is a topic for significant future work. Examples of sentences that use combinations of descriptive adjectives are shown in Figure 10.

Push the large tall chair around the small pretty table
`(push tall_large_chair pretty_small_table around)`

Move the tool above the table
`(move tool table above)`

Put the small yellow book under the brown round table
`(put yellow_small_book brown_table under)`

Move the sharp tool near the tiny red box
`(move sharp_tool red_tiny_box near)`

Move the big purple ball from the tiny red table
`(move purple_big_ball red_tiny_table from)`

Chase the big man to the right door
`(chase big_man right_door to)`

Fig. 10. Sample sentences using prepositions.

3) Use of Prepositions: Given the nature of verbal instructions that we are interested in providing, the focus is on providing location information regarding placing or positioning of objects for a task. Our system can handle prepositions related to locations, such as those in the following list:

```
with, at, from, into, during, against,
among, towards, upon, in, on, by, over,
through, of, throughout, to, for, about,
after, under, within, aboard, next to,
in front of, along, across, behind,
beyond, but, up, out of, out, around,
down, off, above, near, below, beside,
beyond, inside, onto, opposite, outside,
underneath, unto, adjacent to, ahead of,
as of, other than, outside of, as far as
```

Figure 10 shows sentences that use such prepositions.

V. Future Work

We first plan to eliminate the distinction between the basic and high-level tasks, as it is irrelevant to the learning process. We also aim to allow for additional execution constraints while learning multi-instruction tasks (in addition to sequencing) under a parent *THEN* node, which is currently assumed. This can be done by allowing the user to use additional verbal cues prior to each individual instruction.

To take advantage of the richness provided by natural instruction, we plan to extend the capabilities of our parsing algorithm to handle a wider range of prepositions and adjectives, as well as to extend the library of robot behaviors in order to perform more varied tasks. We will also refine the process that maps location information from prepositions to parameter values for the robot's behaviors.

VI. Conclusion

This paper described a novel approach to transfer complex task knowledge from a human user to a robot, with the goal of exploiting the richness of natural language instructions in order to increase the complexity of task representations that a robot can learn. In particular, the focus was on learning tasks with *convey complex execution constraints* (such as alternative paths of execution, sequential or non-ordering constraints, as well as hierarchical representations), as well as on enabling behavior parameterization through the instruction. Specific algorithms have been developed for handling *conjuctions*, *adjectives* and *prepositions* as well as for translating the parsed instructions into parameterized executable task representations. The method also allows to learn increasingly complex tasks from multiple instructions. Experimental validation using a PR2 humanoid robot has been performed, demonstrating the feasibility of the proposed method to learn multiple task representations with complex constraints. Additionally, examples of parsed trees outside of the robot domain are provided in order to demonstrate the versatility of the method.

ACKNOWLEDGMENT

This work has been supported by Office of Naval Research Award N00014-16-1-2312.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2008.10.024

[2] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, “Mobile robot programming using natural language,” *Robotics and Autonomous Systems*, vol. 38, no. 3, pp. 171 – 181, 2002, advances in Robot Skill Learning. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889002001665

[3] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, *Learning to Parse Natural Language Commands to a Robot Control System*. Heidelberg: Springer International Publishing, 2013, pp. 403–415.

[4] F. Duvalet, “Natural language direction following for robots in unstructured unknown environments,” Ph.D. dissertation, Carnegie Mellon University, 2012.

[5] T. C. Lueth, T. Laengle, G. Herzog, E. Stopp, and U. Rembold, “Kantra-human-machine interaction for intelligent robots using natural language,” in *Proceedings of 1994 3rd IEEE International Workshop on Robot and Human Communication*, Jul 1994, pp. 106–111.

[6] M. Ralph and M. A. Moussa, “Toward a natural language interface for transferring grasping skills to robots,” *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 468–475, April 2008.

[7] Y. Bisk, D. Yuret, and D. Marcu, “Natural language communication with robots,” in *Proceedings North American Chapter of the Association for Computational Linguistics*, 2016.

[8] G.-H. Wang, P. Jiang, and Z.-R. Feng, “Extraction of robot primitive control rules from natural language instructions,” *International Journal of Automation and Computing*, vol. 3, no. 3, pp. 282–290, Jul 2006. [Online]. Available: https://doi.org/10.1007/s11633-006-0282-7

[9] J. Thomason, S. Zhang, R. Mooney, and P. Stone, “Learning to interpret natural language commands through human-robot dialog,” in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI’15. AAAI Press, 2015, pp. 1923–1929. [Online]. Available: http://dl.acm.org/citation.cfm?id=2832415.2832516

[10] A. R. Chang, “User-extensible natural language spoken interfaces for environment and device control,” Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2008. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-162.html

[11] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. S. Wong, and S. Tellex, “Accurately and efficiently interpreting human-robot instructions of varying granularities,” *CoRR*, vol. abs/1704.06616, 2017. [Online]. Available: http://arxiv.org/abs/1704.06616

[12] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, “Gated-attention architectures for task-oriented language grounding,” *CoRR*, vol. abs/1706.07230, 2017. [Online]. Available: http://arxiv.org/abs/1706.07230

[13] D. K. Misra, J. Langford, and Y. Artzi, “Mapping instructions and visual observations to actions with reinforcement learning,” *CoRR*, vol. abs/1704.08795, 2017. [Online]. Available: http://arxiv.org/abs/1704.08795

[14] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama, “From language to goals: Inverse reinforcement learning for vision-based instruction following,” *CoRR*, vol. abs/1902.07742, 2019. [Online]. Available: http://arxiv.org/abs/1902.07742

[15] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine, “Guiding policies with language via meta-learning,” *CoRR*, vol. abs/1811.07882, 2018. [Online]. Available: http://arxiv.org/abs/1811.07882

[16] M. Scheutz, E. Krause, B. Oosterveld, T. Frasca, and R. Platt, “Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture,” in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS ’17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1378–1386. [Online]. Available: http://dl.acm.org/citation.cfm?id=3091282.3091315

[17] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang, “Grounding english commands to reward functions,” in *Robotics: Science and Systems*, 2015.

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

[19] D. K. Misra, J. Sung, K. Lee, and A. Saxena, “Tell me dave: Context-sensitive grounding of natural language to manipulation instructions,” *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 281–300, 2016. [Online]. Available: https://doi.org/10.1177/0278364915602060

[20] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, “Understanding natural language commands for robotic navigation and mobile manipulation,” in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, CA, August 2011, pp. 1507–1514.

[21] W. Packard. (2015) Answer constraint engine. [Online]. Available: http://sweaglesw.org/linguistics/ace/

[22] J. Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu, “Language to action: Towards interactive task learning with physical agents.”

[23] M. Forbes, R. P. N. Rao, L. Zettlemoyer, and M. Cakmak, “Robot programming by demonstration with situated spatial language understanding,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2014–2020.

[24] M. Tenorth, D. Nyga, and M. Beetz, “Understanding and executing instructions for everyday manipulation tasks from the world wide web,” in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 1486–1491.

[25] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, “Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction,” *International Journal of Social Robotics*, vol. 4, no. 2, pp. 181–199, Apr 2012. [Online]. Available: https://doi.org/10.1007/s12369-011-0123-x

[26] L. Fraser, B. Rekabdar, M. Nicolescu, M. Nicolescu, D. Feil-Seifer, and G. Bebis, “A compact task representation for hierarchical robot control,” in *International Conference on Humanoid Robots*. Cancun, Mexico: IEEE, November 2016, pp. 697–704.

[27] R. C. Arkin, *An Behavior-based Robotics*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[28] D. Flickinger. (2013) English resource grammar. [Online]. Available: http://www.delph-in.net/erg/

[29] A. Copestake, D. Flickinger, C. Pollard, and I. A. Sag, “Minimal recursion semantics: An introduction,” *Research on Language and Computation*, vol. 3, no. 2, pp. 281–332, Jul 2005. [Online]. Available: https://doi.org/10.1007/s11168-006-6327-9