# Spoken Instruction-Based One-Shot Object and Action Learning in a Cognitive Robotic Architecture

Matthias Scheutz
HRI Laboratory
Computer Science
Tufts University
Medford, MA 02111, USA
matthias.scheutz@tufts.edu

Evan Krause
HRI Laboratory
Computer Science
Tufts University
Medford, MA 02111, USA
evan.krause@tufts.edu

Brad Oosterveld
HRI Laboratory
Computer Science
Tufts University
Medford, MA 02111, USA
bradley.oosterveld@gmail.com

Tyler Frasca
HRI Laboratory
Computer Science
Tufts University
Medford, MA 02111, USA
tmfrasca@gmail.com

Robert Platt
College of Computer and
Information Science
Northeastern University
Boston 02115, USA
rplatt@ccs.neu.edu

## ABSTRACT

Learning new knowledge from single instructions and being able to apply it immediately is a highly desirable capability for artificial agents. We provide the first demonstration of spoken instruction-based one-shot object and action learning in a cognitive robotic architecture and discuss the modifications to several architectural components required to enable such fast learning, demonstrating the new capabilities on two different fully autonomous robots.

## CCS Concepts

•**Human-centered computing** → **Natural language interfaces;**
•**Computing methodologies** → **Online learning settings;**

## Keywords

One-shot learning; natural language instruction; learning actions and objects

## 1. INTRODUCTION

Learning from natural language instructions is undoubtedly among the most impressive feats of the human cognitive system. It starts when children have acquired enough language to understand instructions containing new words, and continues through various developmental phases into adulthood where complex concepts that could otherwise not be experienced can be conveyed merely through natural language descriptions (think of large infinite cardinal numbers $\aleph_\alpha$). It is clear that such a learning capability would be of immense utility to artificial agents. Not only could agents quickly acquire new knowledge, possibly in "one-shot" from a single instruction, but they could also share this newly acquired knowledge with other agents of the same ilk (i.e., if their architectures represent it in a way that allows for knowledge sharing), enabling massive parallel knowledge acquisition among a cohort of agents.

**Appears in:** *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*


While most artificial cognitive systems have the ability to acquire new knowledge, some even from natural language instructions (e.g., [13, 19]), the way new knowledge is acquired in these architectures is not comparable to the human case. For one, cognitive architectures make various assumptions about perceptual and actuation capabilities (i.e., what primitive percepts and actions are available) as well as internal representations (e.g., what concepts and relations can be represented). They typically cannot accommodate truly novel objects or actions, or even novel object parts or known actions performed on those parts. Some of these representational and functional assumptions made by cognitive architectures carry over to cognitive robotic architectures (CRAs), e.g., to task and motion planners which operate on given planning models that consist of given predicates and relations, even though CRAs are better equipped to acquire new primitive percepts or actions. In learning new knowledge from instructions, however, an architecture also has to cope with unknown words in the instruction, in addition to the unknown concepts denoted by those words. This, in turn, requires the natural language subsystem (NLS) of the architecture to be able to cope with all aspects of unknown words: from their acoustic features, to their syntactic properties, to their semantic meaning, and possibly their pragmatic implictures. Hence, to be able to *truly learn from natural language instructions* as humans do, CRAs need to allow for systematic representations of unknown entities such as words, percepts, actions, etc. that can be processed in almost every component of the architecture and subsequently refined based on the semantics of the natural language instructions (and possibly perceptual and other constraining contextual factors). This, in turn, requires modifications to the component algorithms to handle representations that are partial and underspecified. And it requires deep interactions between the NLS and other components in the architecture to allow for subsequent refinement and further specification of those initially incomplete representations.

We tackle the problem of making a cognitive robotic architecture fit for spoken instruction-based learning of objects and actions in *one shot*, i.e., from a single instruction, in such a way that (1) the acquired knowledge about objects and actions is integrated within the existing knowledge, (2) the knowledge can be used immediately by the learning agent for task performance, and (3) the knowledge can be shared immediately with other agents using the same architecture. One-shot learning can be triggered either directly by ex-

plicit statements such as "I will teach you how to do X" or "Look at the object in front of you." followed by an explicit definition such as "First, do A, then B..." or "This is a Y", or indirectly by using an instruction that contains a word the agent does not know "Pick up the knife by the blade" which will prompt it to ask for its definition ("What part of the knife is the blade?"). To guide the exposition, we use a running example, that of a robot which is first taught what the handle of a knife is and then subsequently asked to pick up a knife by its handle. Although nothing hinges on this particular example, it will be useful in the discussion of the various component algorithms, their operation, and the modifications that were necessitated in order to allow for one-shot learning, which follow next.

## 2. INSTRUCTION-BASED ONE-SHOT OBJECT AND ACTION LEARNING

Instruction-based one-shot object and action learning can be defined as learning *conceptual definitions* for objects and actions as well as their aspects (e.g., object parts and action parameters) from natural language expressions that contain these definitions. For example, an object definition such as "A medical kit is a white box with red cross on it and a handle on top" defines a medial kit in terms of other shape, color, and object concepts referred to by color adjectives (e.g., "white" and "red"), shape nouns (e.g., "box" and "cross") and other object types (e.g., "handle") as well as relational expressions such as "on" and "on top" which relate the various object parts. A vision system that knows how to recognize and determine the various ingredients used in the definition can then recognize the new object by way of recognizing its constituent parts and their relationships (cp. to [14]). Similarly, a definition such as "To follow means to stay within one meter of me", again assuming that all concepts after "means" are known, should allow an action execution component to construct an action script that captures the procedural meaning of the expression (cp. to [2]).

**Definition**[Natural language object and action definition]. Let $W_O$ be a set of natural language expressions denoting objects, object properties, or object parts in a set $O$, $W_r$ be a set of relation expressions denoting relations in $R$ among object parts $O$, and $W_v$ be a set of natural language expressions denoting actions and action sequences as well as action modifications in a set of actions $V$. Then the natural language expression $U(w, W_o, W_r, W_v)$ is a *definition* of a concept denoted by $w$ if $U$ contains $w$ in a way that marks $w$ as the *definiendum* (i.e., $w$ is used to denote what is being defined such as saying in "A table is..." or "I will teach you how to pick up...") and the rest of the $U$, the *definiens*, involves any number of expressions from $W_o$, $W_r$, and $W_v$ in a compositional fashion (e.g., "white box with red cross on it" or "stay within one meter") such that the composite meaning denoting a new object or action can be determined from the meaning of its parts.

One-shot object and action learning then amounts to (1) learning the linguistic aspects of the definiendum $w$, (2) determining the semantics of the definiens $U$, and (3) associating the constituent parts of the definiens $U$ with different data representations in the CRA. Assuming that CRA has all functional representations and processes for all $W_o$, $W_r$, $W_v$ (e.g., object, object part and relation detectors in the vision system and action primitives and parameters in the action execution system), then invoking $w$ in subsequent utterances will lead to retrieval and application of these data structures. In other words, by being able to understand natural language definitions of concepts cast in terms of either known concepts or other unknown concepts that are themselves defined in natural language, a CRA can quickly acquire new knowledge by combining existing knowledge in a way prescribed by those definitions.

To make this also practically possible, several changes and additions must be made to a CRA to enable the architecture to handle new words, generate novel data structures based on expressions that refer to other knowledge, and associate those data structures in ways that future invocations of the word will trigger the right kind of retrieval and application processes of the knowledge. For example, the speech recognizer must learn the acoustic signature of the new word on its first occurrence and be able to recognize it subsequently, the syntactic and semantic parsers have to be able to assign a grammatical type, syntactic structure and descriptive semantics to the word, and depending on whether it denotes an action or object, the new term has to be associated with knowledge in the vision and action components. Moreover, the CRA has to detect when new knowledge is presented and understand from the utterance what type of knowledge it is. In the following, we will show in some detail how this can be accomplished in a CRA in way that the NLS in conjunction with other components in the CRA can process, store, retrieve, and apply the defined concepts.

## 3. ARCHITECTURAL MODIFICATIONS

As mentioned above, almost all components in a CRA are implicated in instruction-based one-shot learning and the goal of this section is to show what kinds of modifications are necessary to the various component algorithms to allow them to handle unknown words and concepts as part of the learning process. We will use as a simple guiding example the instruction "Pick up the knife by the handle." given to a robot that does not know what a handle is. Currently, most CRAs would simply fail, either because the speech recognizer does not understand the word; or if the word were in the vocabulary of the recognizer (which it should not be if it is truly considered to be an unknown word), the parser will likely not know what do to with it because it has no part-of-speech tag for the word and thus does not know what grammar rules apply; or if it does (again because the robot has already partial knowledge of the word), the semantics are unknown; or if the semantics are known, then the connection to the perceptual system are unknown, etc. Current instruction-based learning architectures make varying assumptions about what aspects of the unknown word and concepts have to be assumed in order to be able to learn other aspects. No current system, however, is able to learn *all aspects* in one shot.

Before we consider the architectural modifications, it is useful to take a step back and consider how difficult this problem is. First, how can the robot even understand the instruction when it does not know the word "handle", i.e., how can it recognize the word, determine its syntactic features, and then infer its semantic role in the instruction? How can it learn enough of the semantics of "handle" to configure its vision system to be able to perceive the relevant part of the object? And how can it perform the particular pick-up action on the object part when it has never seen the object before and does not know how to best grasp it for the pick-up?

We will address these questions one by one in the subsequent sections in the context of the particular CRA we chose for the project, the DIARC architecture (e.g., [26, 25]), although the modifications discussed below would equally apply to other CRAs. Fig. 1 shows the relevant parts of DIARC that must be adapted in order to allow for object and action-based one-shot learning (additional components might be required for learning additional more abstract concepts). In particular, we will in the following describe the modifications to the algorithms in four of these components which are typically not part of classical cognitive architectures: the Automatic Speech Recognizer, the Parser, the Vision component,

the Action Manager, and the Text-to-Speech component, and show how they are now able to systematically handle partial and incomplete information (we have to leave discussions of how to modify other components for another occasion for space reasons).

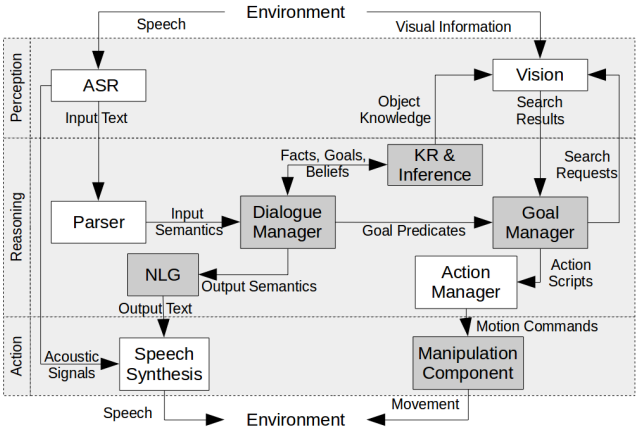


**Figure 1: Architecture overview with relevant components and their information flow (components discussed below are shown in white).**

## 3.1 Speech recognition

The role of the Automatic Speech Recognition (ASR) system is to convert acoustic speech signals into a textual representation. While standard speech recognizers are typically able to detect when a word is not in their vocabulary, they are not able to add it to their vocabulary on the fly so that, ideally, it can be recognized the next time it occurs. We thus developed a special one-shot speech recognizer (OSSR) (which can be used independently or in conjunction with existing large vocabulary speech recognizers) to appropriately detect out-of-vocabulary utterances and learn them in one shot using a nearest neighbor classifier. While not as accurate as more sophisticated classifiers used in ASR, it has the distinct advantage that new classes, and their representations, can be added in very small constant time. Another trade-off is that the cost of $O(1)$ for category insertion comes with the $O(n)$ cost for search. Fortunately, these comparisons are all independent from each other and can thus be parallelized. The implementation used in this evaluation can leverage the available hardware of its system, CPUs or GPUs, to divide that $O(n)$ search time by the number of available threads available (details of the recognizer are described in [21]).

Another critical feature is how the similarity between two sound tokens is measured by the recognizer. We use a version of the Acoustic DP-Ngram Algorithm (DP-Ngrams) [1] modified for the OSSR task. DP-Ngrams is able to not only distinguish when two feature sequences are similar, but also to determine which parts of those sequences are similar, and which are not. This allows for words learned on the fly to be recognized compositionally as parts of a longer utterance.

Previously, DP-Ngrams has been used to derive subword units from sets of similar words [1]. We modified the algorithm weights to favor longer subsequence alignments to focus on the discovery of word level units in input acoustic signals. We also smoothed the input feature sequences before comparison to reduce the effects that minor variations have on alignment similarities.

If an input does not have any neighbors above a similarity threshold it is determined to be a new word. A representation of that word is generated which contains: the signal in the OSSR's feature space, the original acoustic signal, and a text label as a reference for the rest of DIARC. The labels of new words are determined by the order in which they are encountered, the first is labeled *UT1* (for "unknown token 1") and so on. Future instances of words learned on the fly are also stored. In the case of the utterance "Pick up the knife by the handle" where the representation of "handle", acoustic and textual, is not previously known, the ASR component would produce the text "Pick up the knife by the UT1".

## 3.2 Syntactic and semantic parsing

The parser's job is to generate both the syntactic structure and the semantic interpretation of an utterance. This can be done either in two steps – syntactic followed by semantic parsing – or in one step together. The particular parser employed in DIARC is an incremental and extended version of the Combinatory Categorial Grammar (CCG) parser from [6]. It contains a dictionary of parsing rules each composed of three parts: a lexical entry, a syntactic CCG definition of the semantic type of the lexical entry, and the semantics of the lexical entry in lambda calculus. Table 1 shows the rules used for parsing the utterance *"Pickup the knife by the UT1"*.[1]

| Label | Syntax | Semantics |
|---|---|---|
| pickup | C/NP[PP] | $\lambda x.pickup(?ACTOR, x)$ |
| the | NP/N | $\lambda x.x$ |
| knife | N | $knife$ |
| by | (NP[PP]/NP)\NP | $\lambda x \lambda y.partOf(x, y)$ |

**Table 1: A subset of the rules used by the parser.**

The parser stores information about the state of the parse as a set of binary trees. Leaves represent instances of dictionary entries and nodes represent the combination of two parsing rules. Input is received incrementally, word by word, and as words get added to the parse, the parser examines the syntactic rule of the root of all existing trees in the parse space. If the new word can be combined with an existing root, then the relevant trees are updated recursively. If not, the new word is added as the root of a new tree. The parser checks after each update whether the root of one of the parse trees is of a terminal type, in which case the combined semantics are generated from the tree and the parse space is cleared.

Since novel words like "UT1" obviously do not have entries in the parser's dictionary, new entries must be created for them, for which semantic and syntactic definitions must be inferred, which in some cases can be derived from the context in which the word was used (e.g., see [2]). Specifically, the parser currently employs the following heuristic: first, it checks whether the most recent root has unresolved arguments (e.g., "pickup" needs an argument), then it sets the type of the unknown to the argument type required by the previous word. Otherwise, it waits to see whether a new word arrives (within a certain time) and then checks whether the new word expects preceding arguments, in which case it sets the type of the unknown word accordingly, or else it marks the tree as unresolved. If no additional word arrives, it assumes that the unknown word's type must be such that it expects a preceding argument of the type of the preceding root and results in a terminal type.

Once the syntax of the new entry is defined, its formal descriptive semantics can be generated from the label. In cases where the new entry takes no arguments the semantics simply evaluate to the label. When arguments are required, the name of the function in the generated predicate is set to the value of the label. Figure 2

[1]Note that we treat "pickup" as one word in this paper solely to simplify the presentation. In the actual representation, "up" is a modifier for the "pick" action.

shows the parse space before and after the addition of UT1, for the utterance "Pickup the knife by the UT1". The type of the new token is inferred to be "N", and the resultant semantic representation is $pickup(?ACTOR, partOf(knife, UT1))$.

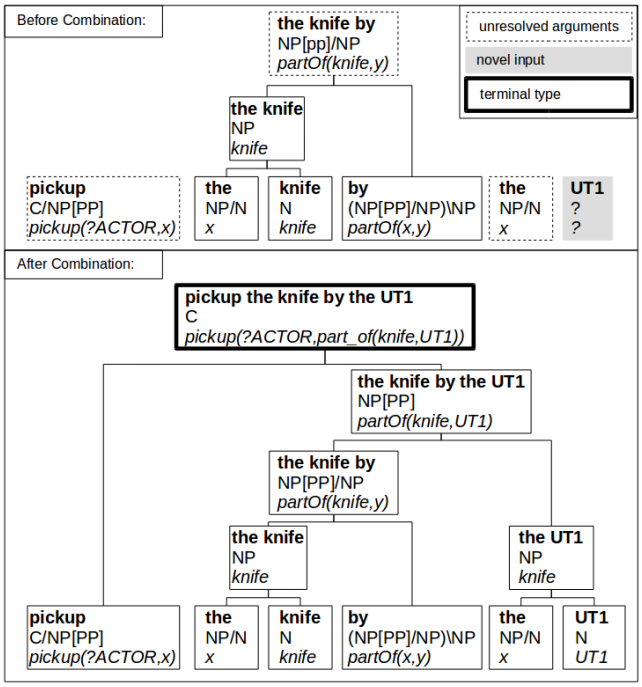


**Figure 2: Parse states before and after *UT1* is added. Labels in bold, semantics italicized.**

## 3.3 Vision processing

The vision system's job is to detect and track objects and their features. It consists of several low-level modules as well as higher-level search managers responsible for configuring and chaining low-level modules into visual searches. The relevant low-level modules in this work include *Detectors*, *Validators*, and *Trackers*, and serve as the building blocks of the system's visual search capabilities. A high-level view of the vision system architecture and the visual search constructed in the proof-of-concept example can be seen in Fig. 3.

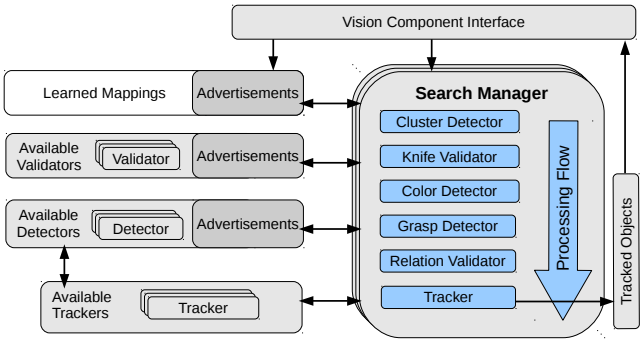


**Figure 3: Simplified vision system overview and the information flow during a visual search (see text for details).**

*Detectors* are responsible for segmenting scenes into candidate objects. Detectors can perform generic segmentation such as clustering objects on a table plane without requiring knowledge of the object types, or more specialized segmentation such as face detection that scans the entire image for a particular object type. Critically, Detectors can also process segmented candidate objects to further segment objects into constituent parts. A search for the "orange part of the knife", for instance, might first employ a knife detection pipeline to detect and segment knives from a scene, and then pass the results to a color detector to detect the orange parts of the detected knives. This flexibility not only prevents the color detector from having to search the entire scene for orange, but also enables the system to build rich object representations by allowing recursive object labeling. *Validators* are similar to specialized Detectors, differing in that they only process pre-segmented candidate objects, and do not scan images, or further segment objects. Classifiers are the most common use case for this type of module, but relational processors (e.g., "part of" relation between two object parts) also fall into this category. *Trackers* are the final stage of the visual search pipeline. After an object has been segmented and labeled as the appropriate target object, it is passed to a tracker to not only track the object from frame to frame, but also make the object available to other components ion the CRA.

To instantiate a new visual search, a Search Manager is created and tasked with taking the requested search descriptors and assembling an appropriate set of modules to satisfy the search constraints. Search requests come in the form of restricted quantifier-free first-order predicate expressions and are then mapped to Validators and Detectors which advertise their capabilities to the vision system via predicate expressions. During the assembly phase, if a Search Manager is unable to find a module to satisfy any part of a search request, the search fails.

To facilitate one-shot learning of new visual concepts through natural language, a vision system must be able to ground new concepts to the real world. This grounding can be achieved in two distinct ways: (L1) mapping new concepts directly to real world object properties, and (L2) mapping new concepts to known concepts. First, to achieve L1, a vision system must be able to segment the object of interest from the scene and extract meaningful features from the object or object part. This approach can be used, for example, when a robot and object are co-located and an utterance such as "this object is a knife" is used to teach a new concept.

To realize L2, a vision system must be able to map new concepts to known concepts in order to ground them to the real world (e.g., see [14]). "The orange part of the knife is the handle", for instance, requires existing knowledge of the "orange", "knife", and "part of" concepts. The vision system presented here, builds internal mappings of these concepts (represented as "Learned Mappings" in Figure3) and is able to instantiate the appropriate module(s) to satisfy requests for newly learned concepts. In this example, a visual search for "handle" would result in the instantiation of modules for "orange", "knife", and "part of" concepts.

Once all parts of a requested predicate expression are satisfied, a search is automatically started and the Search Manager and its modules begin to process captured frames. While performing a visual search, the modules composing the search generate scene graph representations of detected objects. This is a graph structure where nodes represent segmented objects or object parts, and edges represent relationships between nodes (e.g., "part of", "in"). More specifically, nodes contain image masks indicating parts of RGB images and/or RGB-D point clouds, labels indicating properties of the node (e.g., "knife", "orange", "grasp point"), and any supplemental label information (e.g., orientation for grasp points). In addition, each label, on a node or edge, contains a value in [0,1] indicating the confidence of each property.

It is important to note that visual one-shot learning is not capable of learning a generalized concept through a single exposure. In the case of L1, it is impossible to know what features are relevant to the new concept (e.g., color, shape, affordance). Learning that color is not a relevant property of the mug concept, for example, can only be learned through several exposures (or additional NL input). The vision system can only be expected to learn that the object that it currently sees is an instance of the new concept. Similarly for L2, it is only reasonable to assume the mapping holds for the current object. "The orange part of the knife is the handle" clearly does not hold for all handles. Thus, while visual one-shot learning can not create generalized classes of objects and object parts on its own, it does provide a mechanism for both quickly learning instances of new concepts, and also providing labeled training data for generalized concept learning.

**Grasping unknown objects.** Traditional approaches to perception for robotic grasping are object-based [4]. First, they plan grasps with respect to a known mesh model of an object of interest. Then, they estimate the 6-DOF pose of the object and project the grasp configurations planned relative to the object mesh into the base frame of the robot. While these methods can work well in structured environments where object geometries are known ahead of time, they are less well suited for unstructured real world environments. By contrast, grasp detection methods treat perception for grasping like a typical computer vision problem [23, 17, 22, 8, 5, 10, 12, 28, 9]. Instead of attempting to estimate object pose, grasp detection estimates grasp configurations directly from image or point cloud sensor data. These detected grasp configurations are 6-DOF poses of a robotic hand from which a grasp is expected to be successful. Importantly, there is no notion of "object" here and there is no segmentation as a pre-processing step. A successful grasp configuration is simply a hand configuration from which it is expected that the fingers will establish a grasp when closed.

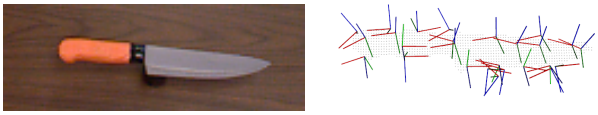

**Figure 4: Left: input object. Right: high scoring grasps on segmented object.**

Most grasp detection methods have the following two steps: grasp candidate generation and grasp scoring. During candidate generation, the algorithm hypothesizes a large number of robotic hand configurations that could potentially be but are not necessarily grasps. Then, during grasp scoring, some form of machine learning is used to rank the candidates according to the likelihood that they will turn out to be grasps. There are a variety of ways of implementing these two steps. These grasp candidates in our approach are generated by searching for hand configurations that are obstacle free with respect to the point cloud and that contain some portion of the cloud between the two fingers. Figure 4 shows a target object and the set of grasps that were scored very highly by the machine learning subsystem. In our case, we use a four-layer-deep convolutional neural network to make grasp predictions based on projections of the portion of the point cloud contained between the fingers. More details on this method can be found in [9].

## 3.4 Action Manager

The Action Manager (AM) in DIARC is responsible for executing action scripts (it corresponds to action monitoring and sequencing component in hybrid robotic architectures). For space reasons, we can only give a high-level overview of the functionality that enables it to learn new action scripts from instructions (possibly spread over multiple dialogue moves). First, the action manager must be notified of an action learning event to be able to start monitoring and recording the instructed actions and action parameters that will have to be assembled into an executable action script. In the case of a direct instruction such as "I will teach you how to do $X$" (e.g., $X$=squat), the AM can infer from the semantics that this is an action or action sequence that pertains to only the actor and does not have any additional actor or object arguments, whereas an instruction such as "I will teach you how to $X$ $Y$" indicates an action plus modifer or object or agent (e.g., $X$=stand $Y$=up, $X$=empty $Y$=the box, or $X$=greet $Y$=John, respectively). Since the NLS will mark $Y$ accordingly, the AM can introduce variables for action parameters, objects, and agents into the signature of the action script, in addition to the actor argument (e.g., `squat(actor:a);` compared to `empty(actor:a,object:y)`). Then, while the to-be-learned actions are instructed, the monitoring process keeps track of all modifications, potentially creating additional typed variables for additional action arguments, as well as state updates to be able to explicitly track any known changes effected by the actions (e.g., after being instructed to "lift your arms", the AM records a state update about the position of the arms). Once the end of the instruction sequence is indicated, the newly formed action script is added to the AM's action database, indexed by the name of the new NL expression denoting it as well as by its initial and goal conditions (to the extent that they were produced during learning). Additional information about the purpose of the action can be added to the goal states if available (e.g., from additional NL instructions). An example of an action scripts is given below in the demonstration section.

## 3.5 Speech generation

Whenever the robot has to communicate with the human, a surface text representation is generated from the logical semantic representations the robot intends to communicate. The speech synthesizer MaryTTS [27], generates synthetic speech based on known mappings between phones and text. In the case of text that represents newly learned words, however, this mapping is not known and the internal representation of learned words is not grounded in a way that can be easily interpreted by humans. This poses difficulty when the robot needs to converse with the human about newly learned topics using newly learned words, it cannot simply ask "What is a UT1?", for example, as "UT1" means nothing to the human. The robot thus not only must be able recognize the acoustic forms of UT1, but it also must produce them. Ideally, the robot would determine some phonological representation of the acoustic signal of UT1, if it corresponds to a word in some language, that could then be used to synthesize it. Since inferring phonological representations from acoustic sigals encoding novel words is not the focus of this project, we chose a simpler route – the "parroting approach" – namely to store the human's acoustic signal originally associated with UT1 and play it back whenever UT1 is to be uttered instead of synthesizing the word using the robot's own speech. While this approach obviously has its shortcomings – aside from sounding funny when the robot plays back the user's voice, it cannot handle inflections or other speech modulations such as stress, prosody, etc. – it is sufficient to demonstrate that the robot can use the newly acquired word token UT1 appropriately in dialogues about establishing its meaning. It is, however, possible for the robot to ask the human to spell the new word (e.g., upon hear-

ing the new word), which will give it a textual representation that it can then use directly to synthesize the new word in its own voice.

# 4. DEMONSTRATIONS

In this section, we briefly walk through two dialogue interactions during which two different robots learns two different types of knowledge from natural language instructions: (1) how to perform a new action sequence (i.e., how to do a squat) and (2) how to recognize and pick up an object by a new part (i.e., the handle of a knife). In both cases, the robot can apply the newly learned knowledge immediately, in the former to do a squat, in the latter to perform the pick-up action on the handle.

## 4.1 Learning Action Sequences in One-Shot

The first demonstration shows how the robot can learn a new action sequence by being instructed to each individual action in sequence:

| | |
|---|---|
| Human: | I will teach how to do a squat. |
| Robot: | OK. |
| Human: | Raise your hands. |
| Robot: | OK. |
| Human: | Crouch. |
| Robot: | OK. |
| Human: | Stand up. |
| Robot: | OK. |
| Human: | Lower your hands. |
| Robot: | OK. |
| Human: | That is how you do a squat. |
| Robot: | OK. |

The semantics of the initial instruction "I will teach you how to do a squat" processed by the Dialogue Manager triggers the monitoring process in the AM that will observe instructions and demonstrations until the end of the learning phase is indicated. When the robot is then instructed to perform a squat:

| | |
|---|---|
| Human: | Do a squat. |
| Robot: | OK. |

the meaning of "squat", the newly learned action script

```
script UT0(actor:a);
  raise-hand(a,all);
  crouch(a);
  stand-up(a);
  lower-hand(a,all);
```

bound to "UT0", gets retrieved and executed in the ActionManager (note that the argument "all" is used since the instruction refers to the plural "hands") . A video of the interaction is located here: `http://bit.ly/2eNVvVW`.

## 4.2 Learning Object Parts in One-Shot

The second demonstration shows how the robot can learn to recognize a new object part and use that knowledge to pick up the object by the newly learned part:

| | |
|---|---|
| Human: | Pick up the knife by the handle. |
| Robot: | OK. |

The ASR recognizes the utterance except for "handle" and creates a new unique identifier "UT1" for it. The recognized text "Pick up the knife by the handle" is sent to the Parser which generates semantics of the form "pickUp(self,partOf(UT1,knife))" and passes it on to the Dialogue system. The utterance is interpreted as a literal command, acknowledged ("OK") and passed on to the KR and Inference component which generates a new goal "pickUp(self,partOf(UT1,knife),leftArm)" that is sent to Action Manager. The Action Manager converts the visual description "partOf(UT1,knife))" to $on(graspPoint, partOf(UT1, knife))$ during the "pickUp" action in order to determine an appropriate grasp point on the object part, and passes it on to the vision system (this is done because the pickUp action requires appropriate grasp points on the object to be detected). Since the vision system does not know what "UT1" is, the visual search fails, and as a result so does the execution of the pick up action. The action failure is communicated to the Dialogue component where a response is generated to address the failure using the recorded sound of "UT1".

| | |
|---|---|
| Robot: | But what is a handle? |
| Human: | The orange part of the knife is the handle. |
| Robot: | OK. |

Here the ASR component recognizes the word "handle" as the same "UT1" token learned from the first utterance, and passes the recognized text to the Parser component where the semantics "is(UT1,partOf(orange,knife))" is generated. The semantic representation is then passed to Dialogue where pragmatic rules modify the semantics form "looksLike(UT1,partOf(orange,knife))" (as this is an instruction about a perceivable object), asserts it into the database of the KR and Inference component, and acknowledges the information ("OK"). This assertion into KR and Inference component triggers a notification to the Vision component which has requested to be notified of all facts of the form "looksLike(X,Y)". It is then able to learn the new grounding of "UT1" to "partOf(orange,knife))". When the robot is then instructed again to pick up the knife by the handle, the same goal as above is asserted, except that the visual search now completes and returns a scene graph of objects to the Action component, which passes it on to the Manipulation component together with the grasp constraints it determined (i.e., similar predicate form of the search request). The Manipulation component then uses the scene graph and grasp constraints to select a grasp point from the subset of grasps satisfying the constraints, and attempts to grasp the object. After the grasp, and additional call to the Manipulation component from the Action Manager is then executed to perform the lift part of the pickUp action. A video of the interaction is located here: `http://bit.ly/2cfx3gL`.

# 5. DISCUSSION

The above walk-throughs show how new, initially meaningless token representations are generated as part of the learning process and become increasingly associated with different meaning representations in different architectural components (the acoustic signal in the ASR, the CCG type and the descriptive semantics in the Parsers, object part descriptions in the Vision system, action sequences in the Action Manager etc.), eventually resulting in full integrated distributed knowledge that is immediately available for use. These are, to our knowledge, the first demonstrations of robots learning how to perform action sequences or how to manipulate unknown parts of known or unknown objects solely from natural language instructions. And, in fact, the robots were able to apply the knowledge in each case immediately, performing the squat or picking up the object properly by the intended part right after having been taught how to do a squat and what the relevant object part looks like, respectively.

Note that above demonstrations also showed that (1) instructions do not have to pertain to a particular set of sensors or actuators and (2) that they do not depend on a particular robotic platform either. Rather, it is possible to include perceptions and perform actions on objects (as in the case of the handle pickup) or to leave out percep-

| Utterance Types | |
|---|---|
| <manipulationCommand> | (<subject>) <manipulation verb> <object> |
| <actionCommand> | (<subject>) <actionVerb> |
| <propertyLearning> | <object> <equality> <objectProperty> <relation> <object> \| <objectProperty> <relation> <object> <equality> <object> |
| <definition> | <unknown> <identity> [<manipulationVerb>\|<actionVerb>\|<objectProperty>\|<object>] |
| <actionTeachingStart> | <startTeachingPrefix> [<actionVerb>\|<manipulationVerb> <object>] |
| <actionTeachingEnd> | <endTeachingPrefix> [<actionVerb>\|<manipulationVerb> <object>] |
| <objectLearning> | <learningPrefix> <object> |
| **Expandable Definitions** | |
| <manipulationVerb> | pick up \| grab \| look for \| find \| give me \| point to \| hand over \| ... |
| <actionVerb> | stop \| start over \| go <direction> \| come here \| follow me \| relax \| [raise\|lower] your hands \| crouch \| stand up \| ... |
| <object> | <object property> <object> \| <object> <relation> <object> \| (a\|an\|the) [knife\|ball\|mug\|box\|...] |
| <objectProperty> | <color> \| <shape> \| <size> \| <texture> \| <part> \| ... |
| **Fixed Definitions** | |
| <relation> | part of \| next to \| on top of \| to the [left \| right] of \| of the \| ... |
| <equality> | is \| ... |
| <identity> | means (the same thing as) \| is the same as \| is like \| ... |
| <actionTeachingStart> | I will [show \| teach] you how to \| this is how [you\|to] \| I will explain how to \|... |
| <actionTeachingEnd> | that is how you \| ... |
| <objectLearning> | [this\|that\|here] is \| [the\|this\|that] object in front of you is \|... |
| <color> | red \| blue \| yellow \| green \| purple \| orange \| black \| white \| gray \| ... |
| <shape> | round \| square \| triangular \| ... |
| <size> | biggest \| smallest \| shortest \| longest \| ... |
| <texture> | shiny \| rough \| checkered \| ... |
| <part> | top \| bottom \| side \| end \| ... |
| <direction> | left \| right \| forward \| backward |

**Table 2: Example set of possible utterances in JSpeech Grammar Format (JSGF).**

tion and perform sequences of actions on various body parts (as in the case of the squat) depending on the platform capabilities. Nor do instructions have to be contained in one sentence (as in the case of the handle), but can be spread over multiple sentences and dialogue interactions (as in the case of the squat). Moreover, learning can be implicitly triggered using a novel word that the robot does not understand (as in the case of the handle) or by explicit initiating the learning interaction (as in the case of the squat). In both cases, new acoustic, syntactic, and formal semantics representations are generated that get associated with the content representations of the instructed knowledge after the relevant parts of the utterances were semantically analyzed (e.g., the visual representations of object part or the action script representation of the action sequence). Hence, together, the discussed architectural components implement the one-shot learning scheme described in Section 2. Moreover, since all knowledge representations in all components (i.e., the associations with the new token learned as part of the learning process in different components) are *purely additive*, i.e., do not modify existing knowledge, it is possible to transmit the knowledge directly to other agents who do not yet have that knowledge for integration into their architectural components (e.g., see [24] for a discussion on how to do this in the middleware used by DIARC).

It is important to point out that the proposed architectural augmentations and the resultant one-shot learning scheme are not limited to the two particular examples demonstrated in the above walkthrough. Rather, being implementations of the general one-shot learning definition in Section 2, they are very general themselves, only limited by the robot's knowledge of natural language as well as its perceptual and actuation capabilities. For example, a robot without legs like the PR2 cannot do a squat even though it can learn how to do it, while a robot without sufficient gripper capabilities like the Nao might not be able to pick up a knife by its handle.

To demonstrate the extent of learning possible in the current implementation, consider Table 2 which provides the grammar (in JSpeech Grammar Format, see `https://www.w3.org/TR/jsgf/`) for the different types of one-shot learning utterance forms that can be handled as well as several terminal expressions (i.e., actual words) that can be used in definitions. For example, the instruction "grab the mug by the handle" is of the form "<manipulationCommand> <object> <relation> <object>". And if the robot does not know what a mug is, asking "What is a mug", it can be instructed "this is a mug" using "<learningPrefix> <object>".

The grammar together with the available object, object part, spatial relation, actions, and action parameters knowledge in the CRA (part of which is indicated in Table 2) can generate infinitely many definitions of new objects and actions (and extending the robots' perceptual and actuation capabilities will further increase the set of possible definitions it can learn). Hence, it is not possible to evaluate the system exhaustively by generating every possible definition and checking whether the architecture has learned the appropriate definition. Nor does it make sense to evaluate a random subset of those expressions, for the same reason that it does not make sense to evaluate a formally correct implementation of the multiplication function by multiplying a subset of numbers: no successful run of any instance can add to the formal correctness of the algorithm. What such instances can show, however, is whether the implementation of the algorithm can practically handle them (and for large numbers we know that this will not be possible). Similarly, teaching the robot large complex definitions of objects and actions will eventually exceed its computational resources and make it fail, even though, in principle, as formalized, it can handle definitions of the kind derivable in the grammar in Table 2.

This raises then the question of how a system like the proposed system which implements a formally correct algorithm (i.e., using meaning expressions in logical definitions cast in natural language to associate the definiendum with the definiens) should then be evaluated. Clearly, empirical runs are important in the robotic case, since implementation details as well as real-time and real-world constraints matter. For this purpose, we provided two uncut videos showing the algorithms at work in real-time on two different fully

autonomous robots. In addition, we provided the grammar of all the utterance forms that can be used to define new object and action concepts that the architecture can then immediately use, making the learning truly one-shot. And the discussion of the NLS showed that the architecture can truly handle new words acoustically, syntactically, and semantically as well on the natural language side.

It is an interesting question to determine the extent to which knowledge acquired through one-shot learning is robust, and is another interesting aspect deserving of further investigation. The two demonstrations discussed above have a nearly 100% success rate when repeatedly instructed after the initial learning instructions (i.e., if the robot is repeatedly instructed to squat or to pick up the knife by the handle). Similarly robust results are obtained using other definitions, but note that ultimately the robustness of application of a newly learned knowledge item depends on the robustness of its constituent parts (e.g., the detectors in the vision system that detect objects and their parts, the action and manipulation algorithms that plan motion parts and carry out action sequences, etc.). Critically, these are not evaluation criteria for one-short learning, but rather evaluation criteria for the learned content and should thus not be conflated with the latter. However, they might be useful in deciding whether knowledge learned quickly through one-shot instructions is sufficiently robust for a task or whether it will have to be altered or augmented to reach the required level of robustness.

There are also interesting open questions about knowledge transfer between robots ensuring that transferred knowledge leads to consistent knowledge bases (because it is still possible, that even though learned knowledge is additive, it could lead to inconsistencies in other systems that do not share exactly the same knowledge bases as the learner), but these will have to be left for another occasion. The important point here is that different from other learning schemes (e.g., neural networks) where new information can alter existing information, the learner itself will remain consistent (to the extent that consistent knowledge is instructed) and can extend its knowledge quickly from a series of instructions.

And, of course, there are many ways to improve different aspects of the current system: the robustness of one-shot speech recognition can be improved (this is a separate research problem on its own), parser grammar type inference and estimation can be extended (to allow for other words and unseen types), allowable utterance forms can be extended, additional primitive actions and object features detectors could be implemented and made available as building blocks in one-shot learning, more natural dialogue moves could be allowed, more complex script learning enabled in the AM, better manipulation actions of novel objects and object parts, proper speech synthesis of newly learned words could be developed, etc. Yet, even without all of these improvements (some of which would be whole research endeavors in their own right), the integrated system demonstrates for the first time a general human-like capability of quickly learning new objects and actions solely from natural language instructions, a capability no other robotic architecture has been able to fully demonstrate (i.e., without taking shortcuts on the NLS, the vision, or the action systems).

# 6. RELATED WORK

While research involving teaching robots through spoken natural language instructions has achieved some successes for both navigation-based tasks [16] and more general tasks [11], as well as through more highly structured dialogues which mimic programming [18], instruction-based one-short learning is still in its infancy. Current approaches to one-short learning are very limited with respect to the allowable teaching inputs and usually can only learn simple behaviors, not complex action sequences (e.g., [2]). And when more complex tasks can be learned through dialogues, additional assumptions are typically made (e.g., the words for the new concepts are already in the speech recognizer, the parser already knows what to do with the word, etc.). Moreover, several of the so-called "one-shot" learning approaches really require multiple trials (e.g., most approaches that focus on visual category, object, and concept learning, in particular, those based on Bayesian approaches, e.g., [7, 15]).

Other work successfully demonstrating robot learning does not use spoken, but rather written instructions. Nyga and Beetz, for example, demonstrated how a robot could learn to follow recipes written in natural language on wikihow.com. Like most research in learning through spoken language, Nyga and Beetz' approach relied on the statistical analysis and use of a variety of corpora (in their case, the WordNet lexical database, the FrameNet action database, the Stanford Parser and wikihow.com, as well as Amazon Mechanical Turk for acquiring labels [20]). The KeJia project has also made progress in allowing robots to learn from written natural-language data [3]; when the OK-KeJia robot detects a gap in its knowledge base (whether conceptual, procedural or functional), it attempts to acquire openly available information to fill the gap (e.g., from the OMICS database).

# 7. CONCLUSION

We presented a general one-shot learning scheme together with modifications to various component representations and algorithms in a cognitive robotic architecture that allow for true one-shot learning of new objects and actions from spoken natural language instructions. Specifically, we demonstrate how the proposed mechanisms allowed different robots to learn how to manipulate an unknown object part after it had received information about the part or execute a newly learned action sequence, respectively. In both cases, after learning, the robots were then able to immediately apply the acquired knowledge. Different from previous work for instruction-based learning, the proposed modifications allow a cognitive robotic architecture to truly acquire new knowledge at every level: from the unknown word and its linguistic properties, to the denoted object concepts and how to manipulate it, to how to perform whole sequences of instructed actions. Moreover, by way of how the newly acquired knowledge is represented and integrated with existing knowledge, it can be shared immediately with other agents running the same architecture.

In a next step, we plan to further improve the various components involved in one-shot learning such as extending the basic manipulation capabilities of the robot beyond grasping objects and thus the range of manipulation behaviors it could learn from natural language instructions, the robot's ability to recognize parts of objects or novel objects, and the natural language understanding capabilities. Moreover, we plan to develop a formal evaluation framework for the performance of one-shot learning architectures, as there is currently not agreed-upon methods for the evaluation of such integrated systems, let alone systems that can quickly acquire new knowledge through natural language instructions.

# 8. ACKNOWLEDGEMENTS

This work has in part been funded by ONR grant #N00014-14-1-0149 and #N00014-14-1-0751 to the first author.

# REFERENCES

[1] G. Aimetti. Modelling early language acquisition skills: Towards a general statistical learning mechanism. In *Proceedings of the 12th Conference of the European Chapter*

*of the Association for Computational Linguistics: Student Research Workshop*, pages 1–9. Association for Computational Linguistics, 2009.

[2] R. Cantrell, P. Schermerhorn, and M. Scheutz. Learning actions from human-robot dialogues. In *Proceedings of the 2011 IEEE Symposium on Robot and Human Interactive Communication*, July 2011.

[3] X. Chen, , J. Xie, J. Ji, and Z. Sui. Toward open knowledge enabling for human-robot interaction. *Journal of Human-Robot Interaction*, 1(2):100–117, 2012.

[4] S. Chitta, E. Jones, M. Ciocarlie, and K. Hsiao. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine*, 19(2), 2012.

[5] R. Detry, C. H. Ek, M. Madry, and D. Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *IEEE International Conference on Robotics and Automation*, 2013.

[6] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA '09)*, Kobe, Japan, May 2009.

[7] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[8] D. Fischinger, M. Vincze, and Y. Jiang. Learning grasps for unknown objects in cluttered scenes. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 609–616. IEEE, 2013.

[9] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt. High precision grasp pose detection in dense clutter. *CoRR*, abs/1603.01564, 2016.

[10] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, T. Asfour, and S. Schaal. Template-based learning of grasp selection. In *IEEE Int'l Conf. on Robotics and Automation*, 2012.

[11] S. B. Huffman and J. E. Laird. Flexibly instructable agents. *arXiv preprint cs/9511101*, 1995.

[12] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4304–4311. IEEE, 2015.

[13] J. Kirk and J. Laird. Interactive task learning for simple games. *Advances in Cognitive Systems*, (3):11–28, 2014.

[14] E. Krause, M. Zillich, T. Williams, and M. Scheutz. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In *Proceedings of Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[15] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Concept learning as motor program induction: A large-scale empirical study. In *Cognitive Science Conference*, 2012.

[16] S. Lauria, G. Bugmann, T. Kyriacou, J. Bos, and A. Klein. Training personal robots using natural language instruction. *Intelligent Systems, IEEE*, 16(5):38–45, 2001.

[17] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. In *Robotics: Science and Systems*, 2013.

[18] C. Meriçli, S. D. Klee, J. Paparian, and M. Veloso. An interactive approach for situated task teaching through verbal instructions. 2013.

[19] S. Mohan, A. Mininger, J. Kirk, and J. E. Laird. Learning grounded language through situated interactive instruction. In *AAAI Fall Symposium Series*, pages 30–37, 2012.

[20] D. Nyga and M. Beetz. Everything robots always wanted to know about housework (but were afraid to ask). In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 243–250. IEEE, 2012.

[21] B. Oosterveld, R. Veale, and M. Scheutz. A parallelized dynamic programming approach to zero resource spoken term discovery. In *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[22] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1316–1322. IEEE, 2015.

[23] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *International Journal of Robotics Research*, 27(4):157, 2008.

[24] M. Scheutz. "teach one, teach all" – the explosive combination of instructible robots connected via cyber systems. In *IEEE Cyber*, 2014.

[25] M. Scheutz, G. Briggs, R. Cantrell, E. Krause, T. Williams, and R. Veale. Novel mechanisms for natural human-robot interactions in the diarc architecture. In *Proceedings of AAAI Workshop on Intelligent Robotic Systems*, 2013.

[26] M. Scheutz, P. Schermerhorn, J. Kramer, and D. Anderson. First steps toward natural human-like HRI. *Autonomous Robots*, 22(4):411–423, May 2007.

[27] M. Schröder, S. Pammi, and O. Türk. Multilingual mary tts participation in the blizzard challenge 2009. In *Proc. Blizzard Challenge*, volume 9, 2009.

[28] A. ten Pas and R. Platt. Using geometry to detect grasp poses in 3d point clouds. In *Proceedings of the International Symposium on Robotics Research*, 2015.