Learning to Parse Natural Language Commands to a Robot Control System

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, Dieter Fox

Abstract As robots become more ubiquitous and capable of performing complex tasks, the importance of enabling untrained users to interact with them has increased. In response, unconstrained natural-language interaction with robots has emerged as a significant research area. We discuss the problem of parsing natural language commands to actions and control structures that can be readily implemented in a robot execution system. Our approach learns a parser based on example pairs of English commands and corresponding control language expressions. We evaluate this approach in the context of following route instructions through an indoor environment, and demonstrate that our system can learn to translate English commands into sequences of desired actions, while correctly capturing the semantic intent of statements involving complex control structures. The procedural nature of our formal representation allows a robot to interpret route instructions online while moving through a previously unknown environment.

1 Motivation and Problem Statement

In this paper, we discuss our work on *grounding* natural language–interpreting human language into semantically informed structures in the context of robotic perception and actuation. To this end, we explore the question of interpreting natural language commands so they can be executed by a robot, specifically in the context of following route instructions through a map.

Natural language (NL) is a rich, intuitive mechanism by which humans can interact with systems around them, offering sufficient signal to support robot task planning. Human route instructions include complex language constructs, which robots must be able to execute without being given a fully specified world model such as a map. Our goal is to investigate whether it is possible to learn a *parser* that produces

· All authors are affiliated with the University of Washington, Seattle, USA.
· Email: {cynthia,eherbst,lsz,fox}@cs.washington.edu

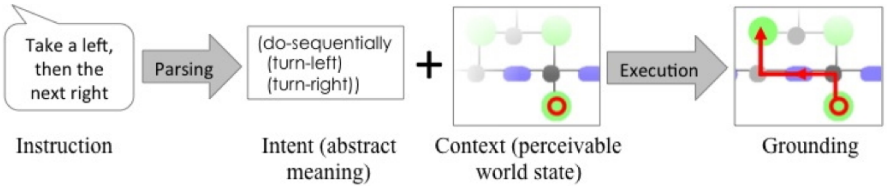


Fig. 1: The task: Going from NL to robot control. First, the natural language command is parsed into a formal, procedural description representing the intent of the person. The robot control commands are then used by the executor, along with the local state of the world, to control the robot, thereby grounding the NL commands into actions while exploring the environment.

correct, robot-executable commands for such instructions. We treat grounding as a problem of parsing from a natural language to a formal control language capable of representing a robot's operation in an environment. Specifically, we train a *semantic parsing* model that defines, for any natural language sentence, a distribution over possible robot control sequences in a LISP-like control language called Robot Control Language, or RCL.

The key contributions of this work are to learn grounding relations from data (rather than predefining a mapping of natural language to actions), and to execute them against a previously unseen world model (in this case, a map of an environment), as illustrated in Fig. 1. Training is performed on English commands annotated with the corresponding robot commands. This parser can then be used to transform new route instructions to execution system inputs in an unfamiliar map. The resulting system can represent control structures and higher-order concepts. We test our approach using a simulator executing the commands produced.

The remainder of this paper is organized as follows. In the next section, we discuss related work in human-robot interaction, natural language understanding, and robot navigation and instruction-following. Sec. 3 describes the technical approach, the formal execution language we define for this work, and our parser learning system. Sec. 4 and Sec. 5 describe the experimental evaluation performed and the results obtained, and we close with a discussion of insights gained from this work.

2 Related Work

Robot navigation is a critical and widely-studied task in mobile robotics, and following natural-language instructions is a key component of natural, multi-modal human/robot interaction. Previous efforts have treated the language grounding task as a problem of parsing commands into formal meaning representations. Several efforts [14, 23] parse natural route instructions to sequences of atomic actions that must be grounded into fully specified world models. Other systems learn to parse navigation instructions, but limit their formal language to a set of predefined parses [25].

Our work falls also into the broader class of grounded language acquisition [24], in which language is learned from situated context, usually by learning over a corpus of parallel language and context data. Other work shows how parsed natural language can be grounded in a robot's world and action models, taking perceptual and grounding uncertainty into account, thereby enabling instruction for robot navigation, GUI interaction, and forklift operation [30, 4, 28].

Parsing natural language to expressive formal representations such as λ-calculus has been demonstrated [31, 1, 18]. λ-calculus is able to represent complex robot control systems [8]; however, to the best of our knowledge, such parser learning approaches have not yet been applied in the context of robotics. Logic-based control systems have been used successfully in robotics [11, 3, 5, 9, 16], providing a powerful framework that can be readily mapped to robot actions, and combinatory categorial grammars have been used for semantic mapping [21]; in contrast to our framework, however, most approaches rely on a manually constructed parser to map from NL commands to λ-calculus, rather than learning grounding relations from data.

Our work is most similar to that of Chen & Mooney [7, 6], who perform parser learning over a body of route instructions through a complex indoor environment containing objects and landmarks with no prior linguistic knowledge. However, their work assumes initial knowledge of a map, and does not represent complex control structures. Compared to our previous approach to parser learning for route instruction following, the system presented here can represent control structures such as 'while,' higher-order concepts such as 'n^{th},' and set operations, and is able to follow such directions through unknown maps.

3 Technical Approach

As noted in Sec. 2, much of the work on learning to parse natural language route instructions assumes previous knowledge of the map to be traversed. Intuitively, our approach is to parse natural language into a high level *specification* of robot behavior, which can then be executed in the *context* of an arbitrary environment. Differentiating context and meaning in this way lends itself well to tasks in robotics, where full knowledge of the environment is rarely available. Our approach also separates the parsing problem from execution, providing a more appropriate layer of abstraction for robot control. This architecture is shown in Fig. 2.

3.1 Robot Control Language

We model control structures in a logic-based Robot Control Language (RCL), inspired by task execution systems such as PRS [12], RPL [2], and GOLEX [11]. For a given set of route instructions, RCL represents the high-level execution intended

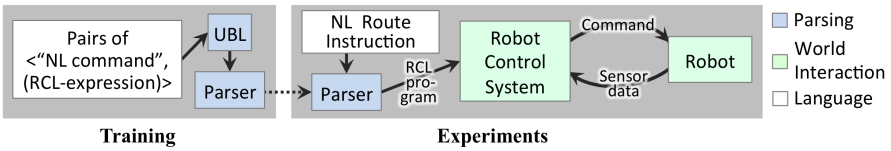


Fig. 2: The high-level architecture of the end-to-end system. Training is performed by learning a parser from English instructions to RCL. In the experimental phase, the learned parser maps NL instructions to an RCL program that is executed by the robot.

by the person. Fig. 3 illustrates RCL programs for paths through an automatically generated, semantically-labeled map [10], along with the instructions corresponding to that path. (RCL is a subset of the typed lambda calculus, and therefore can be represented in a LISP-like format.)

The language is intended to be high-level enough that different execution systems can be used for various tasks, rather than specifying low-level actuation and perception tasks. As such, RCL expressions can encode actions with nontrivial subtasks such as combined perception and action. For example, `(exists left-loc)` is a high-level command to the execution system to determine whether there is an opening to the left, while `(move-to forward-loc)` tells the robot to move one node forward in the semantic map. This also allows the language to be quite compact; a full list of RCL commands is given in Fig. 3(a), including terms for logical expressions, queries about local state, and actions to be performed.

3.2 Parsing

RCL is a formal language defined by a grammar; *parsing* is the process of producing an expression in that grammar from some input. In our case, the input is natural language route instructions, and the parsing target is a statement in RCL that can be passed to a robot controller for planning or further disambiguation.

For this work, parsing is performed using an extended version of the Unification-Based Learner, UBL [17]. The grammatical formalism used by UBL is a probabilistic version of *combinatory categorial grammars*, or CCGs [27], a type of phrase structure grammar. CCGs model both the syntax (language constructs such as NP for noun phrase) and the semantics (expressions in λ-calculus) of a sentence. UBL creates a parser by inducing a probabilistic CCG (PCCG) from a set of training examples.

PCCG-based algorithms have several characteristics that make them a good choice for parsing NL instructions. They are robust against noise found in language, and they are able to efficiently generate n-best parses using CKY-style parsing [29, 18], allowing for jointly considering a parse model and a world model derived from sensor data when interpreting instructions into grounded action; next-best parse search can also be used for "fallback" exploration, e.g., when performing

locations	
`current-loc:loc`	robot's current position
`forward-loc:loc`	location ahead of robot
`left-loc:loc`	to robot's left
`right-loc:loc`	to robot's right
`exists:t [loc]`	does `[loc]` exist?
movement	
`move-to:t [loc]`	move to `[loc]`
`turn-left:t`	take next available left
`turn-right:t`	take next available right
logic	
`and:t [t] [t]`	boolean 'and'
`or:t [t] [t]`	boolean 'or'
`not:t [t]`	boolean 'not'
loops	
`do-until:t [t] [e]`	do `[e]` until `[t]` is true
`do-n-times:t [n] [e]`	do `[e]` `[n]` times
querying the type of a node	
`room:t [loc]`	is `[loc]` a room?
`junction:t [loc]`	is `[loc]` a junction?
`junction3:t [loc]`	is `[loc]` a 3-way junction?
`junction4:t [loc]`	is `[loc]` a 4-way junction?
`hall:t [loc]`	is `[loc]` of type hallway?
mid-level perception	
`turn-unique-corner:t`	take available turn
`take-unique-exit:t`	leave a room with one exit
other	
`<#>:n`	integers
`do-sequentially:t e+`	do each thing in turn
`verify:t t`	error if arg is false

(a)

"Go left to the end of the hall."

```
(do-sequentially
  (turn-left
  (do-until
    (or
      (not
        (exists forward-loc))
      (room forward-loc))
    (move-to forward-loc)))
```

"Go to the third junction and take a right."

```
(do-sequentially
  (do-n-times 3
    (do-sequentially
      (move-to forward-loc
      (do-until
        (junction current-loc
        (move-to
          forward-loc))))
  (turn-right))
```

"Go straight down the hallway past a bunch of rooms until you reach an intersection with a hallway on your left."

```
(do-sequentially
  (do-until
    (and
      (exists left-loc)
      (hall left-loc))
    (move-to forward-loc))
  (turn-left))
```

(b)

Fig. 3: **(a)** The complete list of terms in Robot Control Language. Hallways, rooms and intersections are treated as nodes of a map. The return type of each term is given after its name, followed by the types of any parameters: e (entity), t (boolean), n (number), loc (map location). **(b)** gives examples of English sentences from the test corpus and their RCL interpretations.

local error correction. Additionally, the probabilistic nature of PCCGs offers a clear objective for learning, that is, maximizing the conditional likelihood of training data.

Importantly, UBL can learn a parser solely from training data of the form $\{(x_i, z_i)\}$, where x_i is a natural-language sentence and z_i is a corresponding semantic-language sentence. In brief, UBL learns a model for $p(z_i, y_i | x_i; \theta)$, where θ parameterizes the learned grammar G and y_i is a derivation in G (z_i is completely specified by y_i). UBL uses a log-linear model:

$$p(z_i, y_i | x_i; \theta) \propto e^{\theta \cdot \phi(x_i, y_i, z_i)}$$

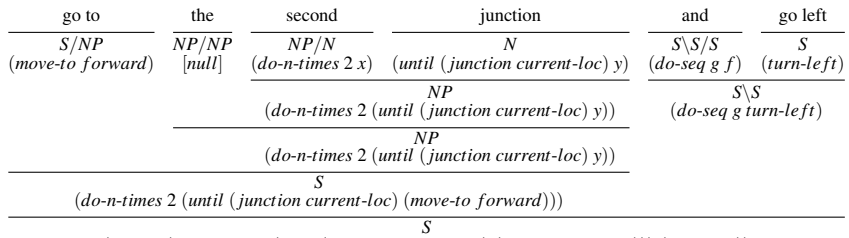


Fig. 4: CCG parse of a test sentence performed by the learned parser. Here the natural language input is first, followed by alternating CCG syntactic categorization and λ-calculus logical forms. The bottom row shows the RCL program to be executed by the robot. (Some syntax has been changed for conciseness.)

UBL first generates a set of possibly useful *lexical items*, made up of natural language words, a λ-calculus expression, and a syntactic category. (An example lexical item might be <"left", `turn-left`, S>.) The algorithm then alternates between increasing the size of this *lexicon* and estimating the parameters of G via stochastic gradient descent [20].

Two types of features θ are used. *Lexical* features fire when the associated lexical items are found in a training example (an example lexical item might be <"left", `turn-left`, S>). *Semantic* features are functions of the logical RCL expression z_i. These are binary features that indicate the co-occurrence of different types of terms in z_i: predicates and their arguments, argument types, predicate co-occurrences, and shared variables. Once a parser is trained, parses are produced via *derivations*, using the learned lexical items and a small set of fixed production rules. Fig. 4 gives an example derivation of an RCL program (last line) from an input sentence (first line).

3.2.1 Parsing Extensions: Initialization of Lexicon and Parameters

In [17], the lexicon—the set of lexical entries and their weights—was initialized with entries covering the entirety of each training example: for each pair of terms found in (x_i, z_i), one initial lexical entry was created. The model defined by θ contained a parameter corresponding to each lexical item, and these weights were set using cooccurrence statistics of single words in the natural language with constants in the semantic language. For example, given the training example:

$$\frac{\text{exit the room and go left}}{(\textit{do-sequentially}\ (\textit{take-unique-exit})\ (\textit{turn-left}))}$$

the algorithm would count one cooccurrence for each of ('exit', *do-sequentially*), ('exit', *take-unique-exit*), ('exit', $turn\text{-}left$), and each other (NL-word, logical-term) pair. The more often such a pair cooccurred, the more important it was considered for parsing and so the higher its weight in the initial model.

In this work we use a new initialization that is better able to handle our semantic language. Intuitively, rather than generating lexical items from each word, we allow arbitrary subsequences of natural-language words, along with semantic subexpressions as defined by the splitting procedure of [17]. As before, this favors (NL, semantics) pairs that are very predictive of each other, while remaining tractable.

More specifically: for each training example i, we let $W(x_i)$ be all subsequences of words in the NL sentence x_i (up to a fixed length, $N = 4$ in our experiments). We define L_i to be the initial *sentential CCG category* of the sentence x_i, having syntactic category S (sentence) and meaning z_i. Splits of L_i into CCG categories are recursively explored to depth two, yielding a set of possible syntactic sub-categorizations R. We then define three count functions: $C(w \in W)$, instances of phrase w in the training data; $C(r \in R)$, occurrences of each syntactic category in the training data; and $C(w,r)$ as the cooccurrence count. The score for lexical entry (w,r) is then defined to be $p(w|r) + p(r|w)$, where probabilities are computed using counts.

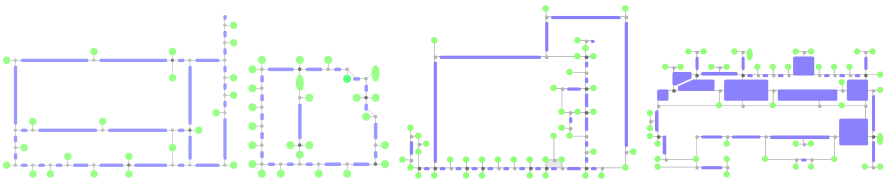
Two other improvements made to UBL involve synonyms and ordinals. During both training and evaluation, if UBL is unable to find a parse for a sentence, it tries to substitute known synonym lists from a standard dictionary for individual words. In addition, numbers are special-cased: when an ordinal (numbers such as 'three', or counting terms such as 'second') is encountered in a training sentence, it is replaced by a standard symbol, turning that training sentence into a *template* into which other ordinals can be substituted freely. As a result, not each ordinal term has to be encountered in all contexts in the training data.

4 Dataset & Maps

Our primary goal is to evaluate the ability of the system to generate RCL command sequences that allow a robot to navigate through a labeled map according to NL instructions. Maps are labeled according to area type (room, hallway, etc.), but are not known to the robot in advance. Instead, the robot explores the map simultaneously with following an RCL program. These experiments are performed in simulation.

We use four maps for our experiments, two each for training and testing (see Fig. 5). Each map has an associated set of routes through the map that have been described in English; for training and test purposes, each English route description is also paired with an associated RCL annotation. Maps A and B—the training and testing maps from earlier work—were automatically generated using Voronoi Random Fields [10] on data gathered by a SICK laser range-finder mounted on a Pioneer robot. Because these original maps were fairly simple, two additional manually constructed maps, C and D, were introduced for training and testing, in order to increase experimental complexity. All navigation experiments were performed in simulated execution of RCL commands in these maps.

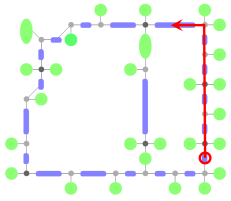
Language datasets were generated as follows. First, English training and testing data from earlier work [23] was re-annotated in RCL; as in [23] and [7], all English instructions are segmented into individual movement phrases. This train-



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

Fig. 5: Four maps were used: A, B, C, and D are respectively a computer science building, an industry research lab, an apartment building, and an academic support building at Cuesta College. Buildings C and D were selected for their relatively complex structure. Areas are color-coded according to type: green areas are rooms, blue areas are hallways, and dark and light gray circles are 4-way and 3-way intersections.

ing set, $\mathbf{S}_{\text{base}}$, contained 189 unique sentences generated by non-experts. We then supplemented this data with additional sentences taken from descriptions of four more complex routes in map C. Twelve non-experts supplied NL directions for those routes. The resulting *enriched* dataset, $\mathbf{S}_{\text{enr}}$, contains 418 NL route instructions along with corresponding RCL command sequences, including structures requiring loops and counting–for example, "Go until you reach a four-way intersection", or "Go into the ninth door on the right". Fig. 6 shows an example of an NL route instruction set with RCL annotation.



(a) Map trace

"Go straight down the hallway past a bunch of rooms until you reach an intersection with a hallway on your left; turn left there."

(b) English phrase

```
(do-sequentially
  (do-until
    (and
      (exists left-loc)
      (hall left-loc))
    (move-to forward-loc))
  (turn-left current-loc))
```

(c) RCL commands

Fig. 6: An example training/testing triplet. (a) The red line shows the path through map B; (b) a description of the path, written by a non-expert person; (c) the language's RCL annotation in $\mathbf{S}_{\text{enr}}$. In testing, the annotation is compared to the annotation produced by the learned parser.

5 Experiments

Experimentally, we are interested in whether the robot reaches the desired destination by exactly following the described path. Since all maps contain loops, any

destination can be reached via an incorrect path; we consider this a failed trial. In addition to the generated RCL program, the robot begins at a known starting location and orientation; local knowledge of the map (adjacent and line-of-sight areas) is updated continuously as the robot progresses. A trial is considered successful only if the robot reached the correct destination along the route intended by the instructor, as our goal is to test instruction-following rather than navigation.

5.1 Baseline, Parser, and Route-Following Experiments

We report on three different experiments used to test the capabilities of our system: two which evaluate its ability to successfully guide a robot through a map, and one which evaluates the parsing component directly. Because the complete number of routes is small and does not necessarily capture map complexity, a large number of test routes of various lengths were generated in the test map, and NL descriptions were generated using route instructions drawn from $\mathbf{S}_{\text{base}}$ and $\mathbf{S}_{\text{enr}}$ (see Fig. 7 for an example test route and description).

5.1.1 Baseline

As a baseline, we test our robot control system on the training, test, and map data used in our previous work, which only contains the 189 comparatively simple route instructions in $\mathbf{S}_{\text{base}}$. For this experiment, data was collected on Map A, and tested against Map B, using the evaluation paradigm from previous work [23]. Performance was virtually identical to that work, with the simulated robot successfully following 71.4% of route instructions. It is not surprising that our new technique did not significantly improve over the performance of the prior approach, as that data set was rather small and did not contain complex control structures (while loops, counting) requiring the additional capabilities provided by this work. Additionally, this accuracy was achieved using only collected training data, without amplifying the training set by adding a large number of hypothetical map routes to each natural language instruction, as was done in [23].

5.1.2 Parser Cross-Validation

In order to evaluate the parser learning system independent of route-following, we perform ten-fold cross-validation on the enriched dataset $\mathbf{S}_{\text{enr}}$, which contains more complex instructions. This test compares the learned parser output against the expert-created RCL annotation, rather than testing the full system against a map, and evaluates performance on individual sentences rather than sets of route instructions. Table 1 shows precision, recall, and F1 score. A parse is reported to be correct if it matches the human-supplied RCL program for that phrase, so this test shows

the frequency with which the parsing system recovers the exact RCL program of unseen commands.

data set	precision	recall	F1 score
enriched	71.0%	72.6%	71.8%

Table 1: Precision, recall, and F1 on cross-validation tests of the extended UBL parser learner.

5.1.3 Route Following with Complex Language

The primary experiment of this work is to test following more difficult route instructions through the complex map D. To determine directly whether this framework can be used to produce executable Robot Control Language, we performed end-to-end tests on a large number of paths through map D in Fig. 5. For each test, $\mathbf{S}_{\text{enr}}$ is split into training and test data by *participant:* all English instructions from some number of non-expert instruction givers ($N = 2$ in our experiments) is withheld from training and used to create a test set $\mathbf{S}_{\text{test}}$. The training set is then $\mathbf{S}_{\text{train}}$=$\mathbf{S}_{\text{enr}}$−$\mathbf{S}_{\text{test}}$, or the set of all sentences collected from the remaining participants.

For each of 10 different sets $\mathbf{S}_{\text{test}1-10}$ (every combination of five different participants), 1,200 paths through map D were randomly generated: 1,000 short paths described by single NL sentences, and 200 paths that required multiple sentences to express (5, on average). An example test path through the map with NL instructions is shown in Fig. 7. The learned parser is asked to interpret these novel route instructions into RCL commands, which were executed by the simulated robot. Table 2 gives a summary of the results.

data set	success (short)	success (long)
enriched	$66.3 \pm 10.7\%$	48.9%

Table 2: Testing the end-to-end system on 1,000 short and 200 longer sets of route instruction. Longer routes average five sentences/route; examples of long and short routes and their associated RCL programs can be seen in Fig. 7 and Fig. 3(b).

5.2 Discussion

Execution of complex language instructions is successful in approximately 66% of short paths, and 49% of complex paths. In general, longer paths are more likely to fail than shorter ones: our simulated control system does not attempt any local

"Go down the long hallway past three intersections, turn left, take the hallway to your left, go through two intersections, turn right, and go forward until you reach the end of the hall."

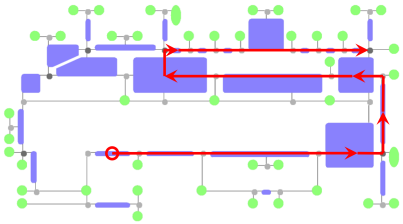


Fig. 7: (left) Example route instruction and (right) corresponding path through map D. Our approach is able to correctly parse 62.5% of such complex instructions into RCL programs that control a simulated robot all the way from the start to the end of such a path.

error recovery if it encounters an unexpected situation, such as discovering that the current map location is not of the expected type or failing to find an opening to the left after a `(turn-left)` instruction. Intuitively, this metric does not give any credit for partly-correct parses of route instructions, e.g., cases where the robot *almost* reaches the destination.

The high variability of results is likely a product of the experimental setup, in which $\mathbf{S}_{\text{test}}$ is generated from statements by people who did not contribute to training data, making the learned parser vulnerable to forms of instruction that are idiosyncratic to an individual. Given that each person contributed route instructions for only four routes (five sentences each, on average), it is also possible that only one or two unparseable instructions occur in a particular data set $\mathbf{S}_{\text{test}}$, such that only randomly-generated paths containing that instruction fail. Both failure modes could be improved by either gathering training data from a wider range of people (for example, using Amazon Mechanical Turk), or by splitting $\mathbf{S}_{\text{test}}$ from $\mathbf{S}_{\text{enr}}$ by *paths*, allowing an individual to contribute route instructions to both training and testing data, but describing different paths or different maps.

Fig. 4 demonstrates the successful derivation of an RCL command from a short test route instruction using our learned parser. As shown, the parser is able to correctly express concepts such as counting ('second') and can generate differing, contextually correct interpretations of the word 'go' (move vs. turn). Fig. 7 shows an example of a long path description that has been successfully parsed by our system.

6 Experimental Insights

Several conclusions can be drawn from these experiments. First, it is possible to learn a parser able to handle complex, procedural natural language commands for robot instruction. This makes it possible to target a rich robot control language capable of handling abstract notions such as counting, loops, and conditionals. We experimentally demonstrate that such a system is able to transform complex natural-language instructions into a language suitable for use by a robot control system.

In this work we demonstrate that it is possible to combine advanced natural language processing with robotic perception and control systems. As an example consider the different, contextually appropriate interpretations of the word 'go' in Fig. 4, where the system learned to interpret 'go to' and 'go left' as having quite different meanings. We established that parsing into a procedural language does require extensions to the parser learning process. We also find that local error recovery would improve results notably, as would a larger, more diverse natural language corpus to train over. In future work, the latter might be obtained from Amazon Mechanical Turk, a system which allows web-based distribution of small tasks to a group of workers in exchange for small payments [13], which it has been used successfully for crowdsourcing natural language data-gathering [26].

This work raises a number of possible directions for future research. It will be worthwhile to see how this system behaves with more complex data that is still drawn from a real robot; for example, using landmarks retrieved from developing object recognition systems [19]. While the effort involved in expert annotation of natural language is far less than that of writing control systems for the tasks, it still requires experts to be involved in the teaching process. Learning solely from execution traces, as in [6], has the potential to increase the efficiency of learning.

We find these results extremely encouraging toward grounding complex NL instructions. The focus of this work is on parsing complex natural language into a formal Robot Control Language. As such, it is complementary to recent efforts in mapping such representations to actions and perceptions in the real world [28, 15, 22]. Since our parser is probabilistic in nature, it can also generate a ranked list of possible RCL programs, each of which could generate a joint model for grounding. We believe that such an approach would enable robots to execute very complex natural language commands.

Acknowledgments

This work was funded in part by the Intel Science and Technology Center for Pervasive Computing, through collaborative participation in the Robotics Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program (Cooperative Agreement W911NF-10-2-0016), and by NSF grant IIS-1115966.

References

1. Y. Artzi and L.S. Zettlemoyer. Bootstrapping semantic parsers from conversations. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2011.
2. M. Beetz, T. Arbuckle, T. Belker, M. Bennewitz, W. Burgard, A.B. Cremers, D. Fox, H. Grosskreutz, D. Hähnel, and D. Schulz. Integrated plan-based control of autonomous service robots in human environments. *IEEE Intelligent Systems*, 16(5), 2001.

3. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2000.
4. S.R.K. Branavan, L. Zettlemoyer, and R. Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *ACL*, pages 1268–1277, 2010.
5. W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.
6. D.L. Chen. Fast online lexicon learning for grounded language acquisition. In *Proc. of the Annual Meetings of the Association for Computational Linguistics (ACL)*, 2012.
7. D.L. Chen and R. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865, 2011.
8. 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 *Proc. of the 2009 IEEE Int'l Conf. on Robotics and Automation (ICRA '09)*, 2009.
9. A. Ferrein and G. Lakemeyer. Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*, 56(11), 2008.
10. S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *IJCAI*, pages 2109–2114, 2007.
11. D. Hähnel, W. Burgard, and G. Lakemeyer. GOLEX - bridging the gap between logic (GOLOG) and a real robot. In *Proc. of the German Conference on Artificial Intelligence (KI), Germany*, 1998.
12. F. Ingrand, R. Chatila, R. Alami, and F. Robert. PRS: A high level supervision and control language for autonomous mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1996.
13. A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proc. of the 26th annual SIGCHI conference on human factors in computing systems*, CHI '08. ACM, 2008.
14. T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *HRI 2010: Proc. of the 5th Int'l Conf. on Human-Robot Interaction*. ACM Press, 2010.
15. H. Kress-Gazit, G. Fainekos, and G. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12), 2008.
16. H. Kress-Gazit, T.Wongpiromsarn, and U. Topcu. Correct, reactive robot control from abstraction and temporal logic specifications. *IEEE Robotics and Automation Magazine, special issue on Formal Methods for Robotics and Automation*, 18(3):65–74, 2011.
17. T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2010.
18. T. Kwiatkowski, L.S. Zettlemoyer, S. Goldwater, and M. Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, 2011.
19. K. Lai, L. Bo, X. Ren, and D. Fox. A scalable tree-based approach for joint object and pose recognition. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.
20. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.
21. P. Lison and G-J. M. Kruijff. An integrated approach to robust processing of situated spoken dialogue. In *Proc. of SRSL 2009, the 2nd Workshop on Semantic Representation of Spoken Language*, pages 58–65, Athens, Greece, March 2009. Association for Computational Linguistics.
22. C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proc. of the 2012 International Conference on Machine Learning*, Edinburgh, Scotland, June 2012.

23. C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010.
24. R. Mooney. Learning to connect language and perception. In Dieter Fox and Carla P. Gomes, editors, *Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence, AAAI 2008*, pages 1598–1601, Chicago, Illinois, 2008. AAAI Press.
25. N. Shimizu and A. Haas. Learning to Follow Navigational Route Instructions. In *Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 2009.
26. R. Snow, B.O'Connor, D. Jurafsky, and A.Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, pages 254–263, 2008.
27. M. Steedman. *The Syntactic Process*. MIT Press, 2000.
28. S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4), 2012.
29. Y. Tsuruoka and J. Tsujii. Iterative cky parsing for probabilistic context-free grammars. *Proc. of the Int'l Joint Conference on Natural Language Processing*, 2005.
30. Y. Wei, E. Brunskill, T. Kollar, and N Roy. Where to go: Interpreting natural directions using global inference. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2009.
31. L.S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence*, 2005.