REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
06-01-2016	Final Report	1-May-2007 - 31-Aug-2014

4. TITLE AND SUBTITLE
Final Report: SUBTLE: Situation Understanding Bot through Language and Environment

5a. CONTRACT NUMBER
W911NF-07-1-0216

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611103

6. AUTHORS
Mitchell Marcus, Hadas Kress-Gazit, Holly Yanco, Daniel Brooks, Constantine Lignos, Cameron Finucane, Kenton Lee, Mikhail Medvedev, Ian Perera, Vasumathi Raman

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Pennsylvania
Office of Research Services
3451 Walnut Street, Suite P-221
Philadelphia, PA 19104 -6205

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
52555-MA-MUR.19

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
While highly constrained language can be used for robot control, robots that can operate as fully autonomous subordinate agents communicating via rich language remain an open challenge. Toward this end, the central goal of the SUBTLE MURI project was to develop an autonomous system that supports natural, continuous interaction with the operator through language before, during, and after mission execution. The operator communicates instructions to the system through natural language and is given feedback on how each instruction was understood

15. SUBJECT TERMS
robotics, natural language, autonomous robots, automatic controller synthesis, formal methods

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mitchell Marcus
UU	UU	UU	UU		19b. TELEPHONE NUMBER
					215-898-2538

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Final Report: SUBTLE: Situation Understanding Bot through Language and Environment

ABSTRACT

While highly constrained language can be used for robot control, robots that can operate as fully autonomous subordinate agents communicating via rich language remain an open challenge. Toward this end, the central goal of the SUBTLE MURI project was to develop an autonomous system that supports natural, continuous interaction with the operator through language before, during, and after mission execution. The operator communicates instructions to the system through natural language and is given feedback on how each instruction was understood as the system constructs a logical representation of its orders using linear temporal logic. While the plan is executed, the operator is updated on relevant progress via language and images and can change the robot's orders. Unlike many other integrated systems of this type, the language interface is built using robust, general purpose parsing and semantics systems that do not rely on domain-specific grammars. The natural language system uses domain-general components that can easily be adapted to cover the vocabulary of new applications. We demonstrate the robustness of the natural language understanding system through a user study where participants interacted with a simulated robot in a search and rescue scenario.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received		Paper
01/04/2013	9.00	João V Graça, Kuzman Ganchev, Luísa Coheur, Fernando Pereira, Ben Taskar. Controlling Complexity in Part-of-Speech Induction, Journal of Artificial Intelligence Research, (07 2011): 527. doi:
01/04/2013	8.00	Funda Durupinar, Nuria Pelechano, Jan Allbeck, Ugur Gudukbay, Norman I. Badler. The Impact of the OCEAN Personality Model on the Perception of Crowds, IEEE Computer Graphics and Applications, (05 2011): 22. doi: 10.1109/MCG.2009.105
01/04/2013	10.00	Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, Ben Taskar. Posterior Sparsity in Unsupervised Dependency Parsing, Journal of Artificial Intelligence Research, (01 2011): 455. doi:
01/04/2016	12.00	Nir Piterman, Vasumathi Raman, Cameron Finucane, Hadas Kress-Gazit. Timing Semantics for Abstraction and Execution of Synthesized High-Level Robot Control, IEEE Transactions on Robotics, (6 2015): 0. doi: 10.1109/TRO.2015.2414134
01/04/2016	14.00	Vasumathi Raman, Cameron Finucane, Mitchell Marcus, Hadas Kress-Gazit, Constantine Lignos. Provably correct reactive control from natural language, Autonomous Robots, (11 2014): 0. doi: 10.1007/s10514-014-9418-8

TOTAL: 5

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received	Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received		Paper
01/04/2013	1.00	Daniel Brooks, Abraham Shultz, , Munjal Desai,, Philip Kovac, Holly A. Yanco. Towards State Summarization for Autonomous Robots, Proceedings of the AAAI Fall Symposium on Dialog with Robots, November 2010. 01-NOV-10, . : ,
01/04/2013	7.00	Vasumathi Raman, Hadas Kress-Gazit. Analyzing Unsynthesizable Specifications for High-LevelRobot Behavior Using LTLMoP, Proceedings of Computer Aided Verification 2011. 14-JUL-11, . : ,
01/04/2013	6.00	Weizi Li, Jan M. Allbeck. Populations with Purpose, Proceedings of Motion in Games. 13-NOV-11, . : ,
01/04/2013	2.00	A. Djalali, D. Clausen, S. Lauer, K. Schultz, C. Potts. Modeling expert effects and common ground using Questions Under Discussion., Proceedings of the AAAI Workshop on Building Representations of Common Ground with Intelligent Agents. 06-NOV-11, . : ,
01/04/2013	3.00	Cameron Finucane, Gangyuan Jing, Hadas Kress-Gazit. LTLMop: Experimenting with Language, Temporal Logic and Robot Control, 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. 12-OCT-10, . : ,
01/04/2013	4.00	Benjamin Johnson, Hadas Kress-Gazit. Probabilistic Analysis of Correctness of High-Level Robot Behavior with Sensor Error, Robotics: Science and Systems . 10-JUN-11, . : ,
01/04/2016	15.00	Vasumathi Raman, └Vasumathi Raman, Constantine Lignos, Cameron Finucane, Kenton C.T. Lee, Mitch Marcus, Hadas Kress-Gazit. Sorry Dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language, Robotics: Science and Systems IX. 24-JUN-13, . : ,
01/04/2016	18.00	Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, Christopher Potts. A computational approach to politeness with application to social factors, Annual Meeting of the Association for Computational Linguistics 2013. 04-AUG-13, . : ,
01/04/2016	17.00	Christopher Potts. Goal-driven answers in the Cards dialogue corpus., 30th West Coast Conference on Formal Linguistics. 13-APR-12, . : ,
01/04/2016	16.00	Daniel J. Brooks, Constantine Lignos, Cameron Finucane, Mikhail S. Medvedev, Ian Perera, Vasumathi Raman, Hadas Kress-Gazit, Mitch Marcus, Holly A. Yanco. Make it So: Continuous, Flexible Natural Language Interaction with an Autonomous Robot, Grounding Language for Physical Systems Workshop at the Twenty-Sixth AAAI Conference on Artificial Intelligence. 22-JUL-12, . : ,
01/04/2016	13.00	Vasumathi Raman, Hadas Kress-Gazit. Automated feedback for unachievable high-level robot behaviors, 2012 IEEE International Conference on Robotics and Automation (ICRA). 14-MAY-12, St Paul, MN, USA. : ,

05/02/2013	5.00	Daniel Lee, Mark McClelland, Joseph Schneider, Tsung-Lin Yang, Dan Gallagher, John Wang, Danelle Shah, Nisar Ahmed, Pete Moran, Brandon Jones, Tung-Sing Leung, Aaron Nathan, Hadas Kress-Gazit, Mark Campbell . Distributed, Collaborative Human-Robotic Networks for Outdoor Experiments in Search, Identify and Track, SPIE Europe Conference on Unmanned/Unattended Sensors and Sensor Networks. 25-OCT-10, . : ,

TOTAL: **12**

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received	Paper

TOTAL:

Number of Manuscripts:

Books

Received	Book

TOTAL:

Received	Book Chapter

TOTAL:

Patents Submitted

Patents Awarded

Awards

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Constantine Lignos	1.00	
Vasumathi Raman	0.50	
Cameron Finucane	1.00	
Dan Brooks	1.00	
Eric Doty	0.10	
FTE Equivalent:	**3.60**	
Total Number:	**5**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Mitchell Marcus	0.16	
Hadas Kress-Gazit	0.08	
Holly Yanco	0.08	
Christopher Potts	0.08	
Jan Allbeck	0.04	
Norman Badler	0.04	
FTE Equivalent:	**0.48**	
Total Number:	**6**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED	Discipline
Kenton Lee	0.10	Computer Science
Israel Geselowitz	0.05	Computer Science
FTE Equivalent:	**0.15**	
Total Number:	**2**	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	2.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	2.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	1.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	1.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	1.00

Names of Personnel receiving masters degrees

NAME

Total Number:

Names of personnel receiving PHDs

NAME
Constantine Lignos
Vasumathi Raman
Munjal Desai

Total Number: 3

Names of other research staff

NAME	PERCENT_SUPPORTED

FTE Equivalent:

Total Number:

Sub Contractors (DD882)

1 a. University of Massachusetts - Lowell	1 b. 600 Suffolk Street, Suite 226
	Lowell MA 018543643
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	5/1/07 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/16 12:00AM

1 a. University of Massachusetts - Lowell	1 b. 450 Aiken Street
	Lowell MA 018543602
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	5/1/07 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/16 12:00AM

1 a. University of Massachusetts - Amherst	1 b. 70 Butterfield Terrace
	Amherst MA 010039242
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	5/1/07 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/15 12:00AM

1 a. University of Massachusetts - Amherst	1 b. Research Administration Building 70 Butterfield Terrace
	Amherst MA 010039242
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	5/1/07 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/15 12:00AM

1 a. Stanford University	1 b. Office of Sponsored Research 3160 Porter Drive, Suite 100			
	Palo Alto	CA	943048445	

Sub Contractor Numbers (c):
Patent Clause Number (d-1):
Patent Date (d-2):
Work Description (e):
Sub Contract Award Date (f-1): 9/1/09 12:00AM
Sub Contract Est Completion Date(f-2): 6/30/16 12:00AM

---

1 a. Stanford University	1 b. 3160 Porter Drive Suite 100		
	Palo Alto	CA	943058445

Sub Contractor Numbers (c):
Patent Clause Number (d-1):
Patent Date (d-2):
Work Description (e):
Sub Contract Award Date (f-1): 9/1/09 12:00AM
Sub Contract Est Completion Date(f-2): 6/30/16 12:00AM

---

1 a. Cornell University	1 b. 171 Kimball Hall		
	Ithaca	NY	14853

Sub Contractor Numbers (c):
Patent Clause Number (d-1):
Patent Date (d-2):
Work Description (e):
Sub Contract Award Date (f-1): 9/1/09 12:00AM
Sub Contract Est Completion Date(f-2): 6/30/13 12:00AM

---

1 a. Cornell University	1 b. 171 Kimball Hall		
	Ithaca	NY	14853

Sub Contractor Numbers (c):
Patent Clause Number (d-1):
Patent Date (d-2):
Work Description (e):
Sub Contract Award Date (f-1): 9/1/09 12:00AM
Sub Contract Est Completion Date(f-2): 6/30/13 12:00AM

---

1 a. George Washington University	1 b. 2121 I Street NW Rice Hall Suite 601 Washington DC 200520086
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	9/1/09 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/12 12:00AM

1 a. George Washington University	1 b. 2121 I Street NW, Suite 601 Washington DC 200520001
Sub Contractor Numbers (c):	
Patent Clause Number (d-1):	
Patent Date (d-2):	
Work Description (e):	
Sub Contract Award Date (f-1):	9/1/09 12:00AM
Sub Contract Est Completion Date(f-2):	6/30/12 12:00AM

Inventions (DD882)

Scientific Progress

See attachment

Technology Transfer

On Wednesday, December 11, 2013, Hadas Kress-Gazit of Cornell, and Mitch Marcus and Constantine Lignos of University of Pennsylvania, conducted a 6 hour workshop at ARL Adelphi for CISD researchers in natural language processing and small robots as a next step towards transitioning the SUBTLE MURI technology to them. About 10 ARL researchers attended. The focus of the workshop was how to extend the SUBTLE natural language lexicon, which contains formula to translate verb semantics to Linear Temporal Logic.

This followed a 4 day “hackathon” by Stuart Young’s small robots group which successfully ported the SUBTLE MURI NLP robot interface to the Packbot platform they were then using – all without our help. The Adelphi group started with the codebase which was transitioned to ARL Aberdeen the previous summer and which we adapted to their software environment.

Situation Understanding Bot using Language and Environment
Final Report
Grant W911NF-07-1-0216

Daniel J. Brooks, Constantine Lignos, Cameron Finucane, Kenton Lee, Mikhail S. Medvedev, Ian Perera, Vasumathi Raman, Hadas Kress-Gazit, Mitch Marcus, Holly A. Yanco

Foreword

While highly constrained language can be used for robot control, robots that can operate as fully autonomous subordinate agents communicating via rich language remain an open challenge. Toward this end, the central goal of the SUBTLE MURI project was to develop an autonomous system that supports natural, continuous interaction with the operator through language before, during, and after mission execution. The operator communicates instructions to the system through natural language and is given feedback on how each instruction was understood as the system constructs a logical representation of its orders using linear temporal logic. While the plan is executed, the operator is updated on relevant progress via language and images and can change the robot's orders. Unlike many other integrated systems of this type, the language interface is built using robust, general purpose parsing and semantics systems that do not rely on domain-specific grammars. The natural language system uses domain-general components that can easily be adapted to cover the vocabulary of new applications. We demonstrate the robustness of the natural language understanding system through a user study where participants interacted with a simulated robot in a search and rescue scenario. Language-enabled autonomous systems of this type represent important progress toward the goal of integrating robots as effective members of human teams.

Keywords Natural language · Formal methods · High-level control · Synthesis

1 Introduction

Robots have the ability to play a unique role in a team of humans in scenarios such as search and rescue where safety is a concern. Traditionally the overhead of interacting with such systems has been high, making integration of robots into human teams difficult. Improvements in natural language technology may, however, allow for this overhead to be reduced if a system can understand natural language input well enough to carry out the required scenario and maintain contact with human team members. In this report we present a system that makes progress towards the use of robots as capable members of a human team by providing continuous, flexible, and grounded natural language communication with the robot throughout execution.

The challenge of programming robots to perform these tasks has until recently been the domain of experts, requiring hard-coded high-level implementations and ad-hoc use of low-level techniques such as path-planning during execution. Recent advances in the application of formal methods for robot control have enabled automated synthesis of correct-by-construction hybrid controllers for complex high-level tasks (e.g., Kloetzer and Belta 2008; Karaman and Frazzoli 2009; Bhatia et al 2010; Bobadilla et al 2011; Kress-Gazit et al 2009; Wongpiromsarn et al 2010). However, most current approaches require the user to provide task specifications in logic or a similarly structured specification language. This forces users to formally reason about system requirements rather than state an intuitive description of the desired outcome. Furthermore, the outcome of verifying such specifications is traditionally simply a response of success or failure; detecting which portions of the specification are at fault is a non-trivial task (e.g., Raman and Kress-Gazit 2013a), as is explaining the problem to the user.

The system presented in this paper combines the power of formal methods with the accessibility of natural language, providing correct-by-construction controllers for specifications that can be implemented and easy-to-understand feedback for those that cannot. The system is open-source, can be extended to cover new scenarios, and allows for both specification and execution of natural language specifications. The system enables users to specify high-level behaviors via natural language, parsing commands using semantic analysis to create a linear temporal logic (LTL) specification. This parsing is deterministic and predictable, providing feedback to the user to help guide them if their input could not be completely parsed. The LTL specification is used to synthesize a hybrid controller when possible. If no implementation exists, the user is provided with an explanation

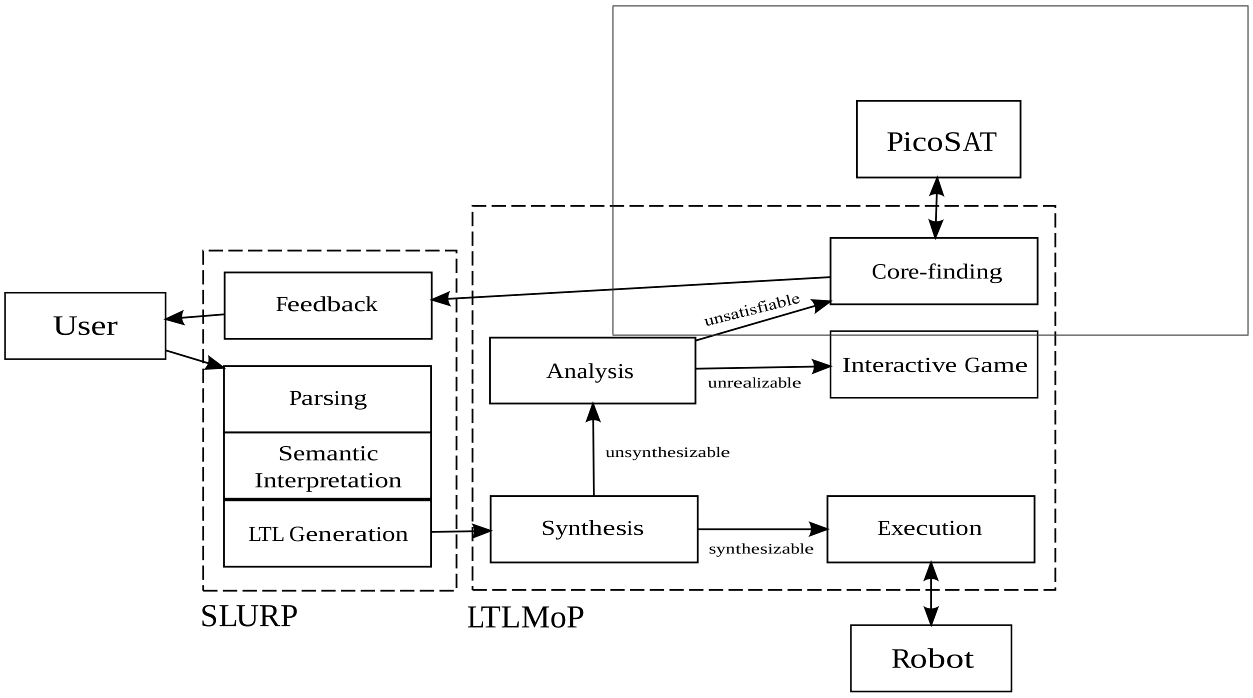


Fig. 1: System overview

and the portions of the specification that cause failure. The explanation is enabled by a data structure called a generation tree, which records the transformation of natural language into formal logic and is able to map backwards from problematic logical propositions to the explicit or implicit input from the user that caused the problem.

The system allows for execution of the controller either in simulation or using a physical robot. Natural language features are also available during execution; the generation tree allows the robot to explain what goal it is trying to achieve through its current action. The same language processing and semantic extraction components that enable the creation of logical propositions from natural language specifications enable feedback from the robot during execution.

The application domain for this system is an urban search and rescue scenario where an autonomous mobile robot acts as part of a team of humans working to explore the area and react to the environment as required. The robot acts as the commander's subordinate, receiving orders and carrying them out. The robot's primary purpose is reconnaissance, entering areas that may be unsafe ahead of human team members. It is assumed that the commander and robot will rarely be colocated. Interaction with the system is implemented through a multimodal tablet interface that acts as a conduit for both sending instructions to the robot and displaying information about the remote situation and environment.

In the sample interaction shown in Figure 2, the system is asked to report when it encounters hostages, instructed to defuse any bombs it finds, and given a set of rooms to search. The system analyzes commands as they are given to it, and when it is told to begin carrying out orders it forms a plan from those commands and begins execution. The robot maintains contact with the operator during execution, informing the operator about anything it was explicitly asked to mention in addition to anything related to the goals it was given.

Commander: Tell me if you see any hostages.
Robot: I'll let you know if I see a hostage.
C: Defuse all the bombs you see.
R: Got it. I'll defuse all bombs.
C: Search the library, classroom, and lab.
R: Got it. I'll search the library, search the classroom, and search the lab.
C: Make it so.
R: Understood. I'm carrying out your orders now.
R: I see a hostage.
R: I'm now going to defuse in the library.
R: I'm done, and I'm in the classroom.

Figure 2: Sample interaction with the system.

Fig. 1 shows the system's main components and the connections between them. The Situated Language Understanding Robot Platform (SLURP) consists of parsing, semantic interpretation, LTL generation, and feedback components. This module is the natural language connection between the user and the logical representations used within the Linear Temporal Logic MissiOn Planning (LTLMoP) toolkit (Finucane et al, 2010). LTLMoP, which also interfaces with the SAT solver PicoSAT (Biere, 2008), provides an environment for creating, analyzing, and executing specifications.

This organization of this paper follows the flow of user input into the system. Sect. 2 discusses previous work in this area. Sect. 3 describes the language understanding and LTL generation mechanisms and presents an evaluation of the performance of the system when processing commands from users. Sects. 4–5 describe controller synthesis and specification feedback mechanisms and present an example of applying the system in a hospital setting. Discussion of the system and future work follows in Sect. 6.

2 Related work

There are many previous approaches that use natural language for controlling robots, differing in the type of information they seek to extract from natural language, the type of dialog desired, and the role of learning. To produce formal goal descriptions and action scripts for tasks like navigation and manipulation, Dzifcak et al (2009) use a combinatorial categorial grammar (CCG) parser with a pre-specified mapping between words or phrases and the matching branching temporal logic and dynamic logic propositions. Another area of focus is language grounding scenarios where language must be mapped to objects or locations (e.g., Matuszek et al, 2012; Tellex et al, 2011), learning the relationship between the words used and the referents in the world. Recent work has focused on the automatic learning of the relationship between language and semantic structures, both in general semantic parsing (e.g., Berant and Liang, 2014; Poon and Domingos, 2009) and robotics-specific applications (e.g., Chen and Mooney, 2011; Matuszek et al, 2010, 2012, 2013). While the learning strategies used may be generalized to broader applications, these systems are typically trained and evaluated in a single command domain (e.g., navigation, manipulation) and typically verifyunderstanding at the per-utterance level or a single action sequence such as sequential navigation commands.

In contrast to learning-focused work, the system presented here focuses on the construction of a complete, formally-verified specification from natural language, but assumes the grounding between language and the robot's capabilities can be specified in advance. In doing so, we explore the broader relationships between natural language and logical form and the challenges involved in creating rich specifications from natural language that can include any number of action types. To be applied to the kinds of high-demand scenarios where natural language control may be of greatest benefit, it is essential that the design of the system be centered around "failing fast" by reporting any errors before execution.

This work aims to generate controllers for autonomous robots that achieve desired high-level behaviors, including reacting to external events and repeated patrol-type behaviors. Examples of such high-level tasks include search and rescue missions and the control of autonomous vehicles following traffic rules in a complex environment, such as in the DARPA Urban Challenge. With the usual approach of hard-coding the high-level aspects and using path-planning and other low-level techniques during execution, it is often not known a priori whether the proposed implementation actually captures the high-level requirements. This motivates the application of formal frameworks to guarantee that the implemented plans will produce the desired behavior.

A number of frameworks have recently been proposed, some of which use model checking (Clarke et al, 1999) to synthesize control laws (e.g., Kloetzer and Belta 2008; Bhatia et al 2010) on a discrete abstraction of the underlying system. Other approaches, such as those proposed by Kress-Gazit et al (2009) and Wongpiromsarn et al (2010), apply efficient synthesis techniques to automatically generate provably-correct, closed-loop, low-level robot controllers that satisfy high-level reactive behaviors specified as LTL formulas. Specifications describe the robot's goals and assumptions regarding the environment it operates in, using a discrete abstraction. The hybrid robot controllers generated represent a rich set of infinite behaviors, and the closed loop system they form is guaranteed to satisfy the desired specification in any admissible environment, one that satisfies the modeled assumptions.

Previous work using LTL synthesis for robot control has also used highly structured or domain-specific languages to allow non-technical users to write robot specifications, even if they are unfamiliar with the underlying logic. For example, LTLMoP includes a parser that automatically translates sentences belonging to a defined grammar into LTL formulas (Kress-Gazit et al, 2008; Finucane et al, 2010); the grammar includes conditionals, goals, safety sentences and non-projective locative prepositions such as between and near. Structured English circumvents the ambiguity and computational challenges associated with natural language, while still providing a more intuitive medium of interaction than LTL. However, users still need to understand many details of the logical representation and the synthesis process to successfully write specifications in structured English. The work presented in this paper replaces LTLMoP's structured English input with a natural language interface to enable users to describe high-level tasks in more natural language.

This paper extends work presented by Raman et al (2013) by analyzing the performance of the natural language understanding system through a user study and adding additional LTL generation features, support for feedback for all types of unsynthesizable specifications, and further discussion regarding the design of the integrated system. We provide a more complete approach to generating LTL from natural language input (Section 3), addressing the issues of non-compositionality when considering negation over natural language and providing results from a user study that demonstrates the system's robustness when used by inexperienced users.

3 Transforming natural language into logic with SLURP

To convert the user's commands into a formal specification, the system must identify the underlying linguistic structure of the commands and convert it into a logical representation, filling in appropriate implicit assumptions about the desired behavior. This section describes the process of this conversion and its implementation as The Situated Language Understanding Robot Platform (SLURP). SLURP enables the conversion of natural language specifications into LTL formulas, communication with the user regarding problems with specifications, and feedback to the user during execution. Sects. 3.2–3.3 discuss the process of generating LTL formulas from natural language input. Sect. 3.4 describes a user study used to evaluate the performance of the system described.

3.1 Overview

In using the term natural language, we refer to language that a user of the system would produce without specific knowledge of what the system is capable of understanding. In other words, language that is not restricted to a known set of vocabulary items or grammatical structures specific to the system. Users are able to give commands without any knowledge of how the language understanding system works, as if they were giving simple, clear instructions to another person. While the user may be able to give commands to the system that it cannot understand, the system gives feedback based on what it understood and what it did not. For example, it may report that it does not understand how to carry out the verb used in a command, or may report that it does know how to perform that verb but has not received sufficient information, for example being told to move but not told where to do so.

The user's instructions are processed through a pipeline of natural language components similar to that used by Brooks et al (2012) which identify the syntactic structure of the sentences, extract semantic information from them, and create logical formulas to be used in controller synthesis. While many previous natural language systems for robot control have relied on per-scenario grammars that combine semantic and syntactic information (e.g., Dzifcak et al 2009), this work uses a combination of robust, general-purpose components for tagging and parsing the input. An advantage of this approach compared to per-scenario grammars is that the core language models need not be modified across scenarios; to adapt to new scenarios all that is required is that the LTL generation be extended to support additional types of commands. This reduces the role of the fragile process of grammar engineering and minimizes the cost of adapting the system to handle commands in new domains.

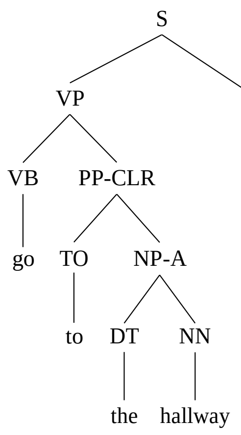
3.2 Identifying linguistic structure

Before a sentence may be converted into logical formulas, the linguistic structure of the sentence must be identified. Following traditional practices in natural language processing, this process is divided into modules: first, the syntactic structure of the input is extracted; second, the meaning of the sentence is recovered by identifying verbs and their arguments. These steps are explained in detail in Sects. 3.2.1 and 3.2.2.

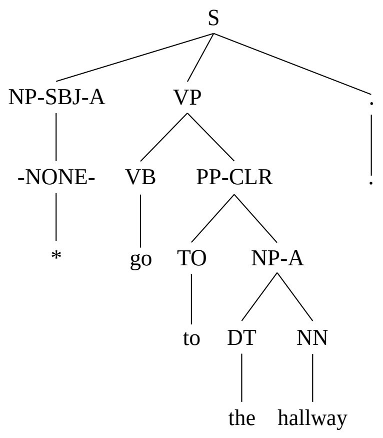
3.2.1 Parsing and null element restoration

Parsing is the process of assigning a hierarchical structure to a sentence. While simple natural language understanding can be performed with shallower processing techniques, parsing allows for recovery of the hierarchical structure of the sentence, allowing for proper handling of natural language phenomena such as negation (e.g., Never go to the lounge) and coordination (e.g., Go to the lounge and kitchen), which are crucial to understanding commands. SLURP uses a pipeline of domain-general natural language processing components. The input is tagged using the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al, 2003) and parsed using the Bikel parser (Bikel, 2004); these parses are then post-processed using the null element (understood subject) restoration system of Gabbard et al (2006). The models used by these systems require no in-domain training. An example output of these modules is given in Fig. 2a, b.

The use of null element restoration, a step typically ignored in NLP systems, allows for correct parsing of imperatives and questions, critical structures for natural language control and dialog systems. For example, in Fig. 2a the original parse contains no subject at all as there is no overt subject in the input sentence. Fig. 2b shows that null element restoration has added an understood subject marked by *. This allows the structure of imperatives to be straightforwardly matched by the semantic interpretation module, which will look for verbs by identifying subtrees that are rooted by S, and contain a subject (NP-SBJ) and a verb phrase (VP). After null element restoration, an imperative has the same underlying structure as a statement with an explicit subject (e.g., A patient is in r1), allowing a general-purpose semantic interpretation system to process all input without using ad hoc techniques to accommodate imperatives.



(a) Tagging and parsing



(b) Null element restoration

Agent: * (understood subject)
Verb: go Preposition:
to Location: the
hallway

(c) VerbNet frame matching

Initially, the hallway has not been visited:
¬s.mem_visit_hallway
Define a persistent memory of going to the hallway:
(s.mem_visit_hallway ⇔
(s.mem_visit_hallway ∨ s.hallway))
Always eventually have a memory of visiting the hallway:
(s.mem_visit_hallway)

(d) LTL formula generation

Fig. 2: Conversion of the sentence Go to the hallway into LTL formulas through tagging, parsing, null element restoration, semantic interpretation, and LTL generation

3.2.2 Semantic interpretation

The semantic interpretation module uses the parse tree to extract verbs and their arguments. For example, in the sentence Carry meals from the kitchen to all patient rooms, the desired structure is a carry command with an object of meals, a source of kitchen, and a destination of all patient rooms. Objects identified may be further processed, for example all will be identified as a quantifier and handled as described in Sect. 3.3.2.

To identify verbs and their arguments in parse trees, SLURP uses VerbNet (Schuler, 2005), a large database of verbs and the types of arguments they can take. The VerbNet database identifies verbs as members of senses: groups of verbs which in similar contexts have similar meanings. For example, the verbs carry, lug, and haul belong to the sense CARRY, because in many contexts they are equivalent in meaning. For each sense, VerbNet provides a set of frames, which indicates the possible arguments to the sense.

In the preceding example sentence, the verb carry is mapped to the sense CARRY. An example of a frame for this sense is (Agent, Verb, Theme, Source, Destination), thus the expected use of the verb is that there is someone performing it (Agent), someone or something it is being performed on (Theme), and path to perform it on (Source and Destination). Each role in the frame, subject to its associated syntactic constraints, is mapped to a part of the parse tree. VerbNet only provides information about the verb arguments; SLURP matches these argument types by mapping them to part-of-speech (e.g., VB for base verb) and phrasal (e.g., NP for noun phrase) tags and identifying each argument using its tag and syntactic position. For the above example, SLURP creates the following mapping:

Agent: the robot (understood subject)
Verb: carry Theme:
meals Source: the
kitchen
Destination: all patient rooms

Among the frames that match the parse tree, SLURP chooses the frame that expresses the most semantic roles. For example, the CARRY sense also contains a frame (Agent, Verb, Theme, Destination), which may be used in cases where the source is already understood, for example if the user previously stated The meals are in the kitchen. However, this frame will not be selected in the above example because the more specific frame that contains a source—and thus expresses more semantic roles—also matches. The chosen match is then used to fill in the appropriate fields in the command.

Matching of frames is not limited to entire sentences. In a sentence such as If you see an intruder, activate your camera, frames are matched for both the conditional clause and the main clause, allowing for the condition (Agent: the robot, Verb: see, Theme: an intruder) to be applied to the action (Agent: the robot, Verb: activate, Theme: your camera) when generating the logical representation.

3.3 Linear temporal logic generation

The information provided by VerbNet allows the identification of verbs and their arguments; these verbs can then be used to generate logical formulas defining robot tasks.

3.3.1 Linear temporal logic

The underlying logical formalism used in this work is linear temporal logic (LTL), a modal logic that includes temporal operators, allowing formulas to specify the truth values of atomic propositions over time. Let $AP = X \cup Y$, where X is the set of "input" propositions, those controlled by the environment, and Y is the set of "output" propositions, those controlled by the robot. LTL formulas are constructed from atomic propositions $\pi \in AP$ according to the following recursive grammar:

$$\phi ::= \pi \mid \neg\phi \mid \phi \vee \phi \mid \quad \phi \mid \phi \; U \; \phi,$$

where $\neg$ is negation, $\vee$ is disjunction, is "next", and U is a strong "until." Conjunction ($\wedge$), implication ($\Rightarrow$), equivalence ($\Leftrightarrow$), "eventually" ( ) and "always" ( ) are derived from these operators. Informally, the formula ϕ expresses that ϕ is true in the next time step. Similarly, a sequence of states satisfies ϕ if ϕ is true in every position of the sequence, and ϕ if ϕ is true at some position of the sequence. Therefore, the formula ϕ is satisfied if ϕ is true infinitely often. For a formal definition of the LTL semantics, see Clarke et al (1999).

Task specifications in this work are expressed as LTL formulas from the fragment known as generalized reactivity of rank 1 (GR(1)), and have the form $\phi = \phi_e \Rightarrow \phi_s$ with $\phi_p = \phi_p^i \wedge \phi_p^t \wedge \phi_p^g$, where ϕ_p^i, ϕ_p^t and ϕ_p^g for $p \in \{e, s\}$ represent the initial conditions, safeties and goals, respectively, for the environment (e) and the robot (s). The restriction to GR(1) is for computational reasons, as described in Kress-Gazit et al (2009).

3.3.2 Types of commands

There are two primary types of properties allowed: safety properties, which guarantee that "something bad never happens," and liveness conditions, which state that "something good (eventually) happens." These correspond naturally to LTL formulas with operators and . While the domain of actions expressible in natural language is effectively infinite, the set of actions that a robot can perform in practice is limited. Examples of semantic behaviors currently implemented include:

1. Actions that need to be completed once, for example going to rooms (Go to the hallway.)
2. Actions that need to be continuously performed, for example patrolling multiple areas (Patrol the hallway and office.)
3. Completing a long-running action that can be interrupted, for example searching a room and reacting to any items found (Search the hallway.)
4. Following (Follow me.)
5. Enabling/disabling actuators (Activate your camera.)
6. Carrying items (Carry meals from the kitchen to all patient rooms.)

Each command is mapped to a set of senses in VerbNet so that a varied set of individual verbs may be used to signify each command. As a result, SLURP is only limited in its vocabulary coverage by the contents of VerbNet—which is easily expanded to support additional verbs if needed—and by what actions can be transformed into LTL. While the number of syntactic structures identifiable by the system is unbounded, the set of frames that SLURP can recognize and transform into logical form is constrained by the mapping of frames to robot actions.

Each command may be freely combined with conditional structures (If you hear an alarm...), negation (Don't go to the lounge), coordination (Go to the hallway and lounge), and quantification (Go to all patient rooms). The use of quantification requires that, in constructing the scenario, information about quantifiable sets is specified; for example, the command "go to all patient rooms" can be unrolled to apply to all rooms that have been tagged with the keyword patient.

3.3.3 Generation

For each supported command, LTL is generated by macros
For each supported command, LTL is generated by macros which create the appropriate assumptions, restrictions, and goals. In the example given in Fig. 2, the resulting LTL formulas define a memory of having visited the hallway, and the goal of setting that memory.[1]

Formulas are generated by mapping each command to combinations of macros. These macros include:

1. Goals: goal(x) generates (x)
2. Persistent memories: memory(x) generates (s.mem_x $\Leftrightarrow$ (s.mem_x $\vee$ s.x)))
3. Complete at least once (ALO): alo(x) generates (goal(s.mem_x) $\wedge$ memory(x))

[1] This arguably unintuitive translation is due to specifications in LTLMoP being restricted to the GR(1) fragment of LTL.

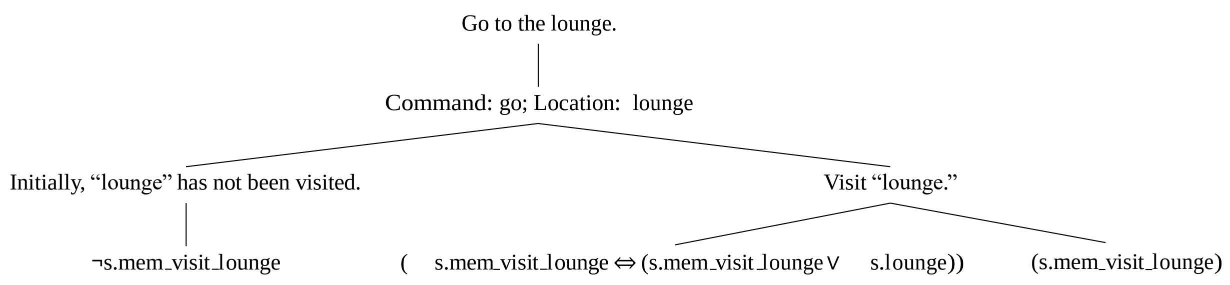

Go to the lounge.
Command: go; Location: lounge
Initially, “lounge” has not been visited.
Visit “lounge.”
¬s.mem_visit_lounge
(s.mem_visit_lounge ⇔ (s.mem_visit_lounge ∨ s.lounge))
(s.mem_visit_lounge)

Fig. 3: Generation tree for Go to the lounge

Among the simplest commands to generate are those that are directly mapped to goals; i.e., ones that are performed infinitely often. For example, patrolling a room maps to goal(room); if multiple rooms are to be patrolled, execution will satisfy each goal in turn, moving the robot from room to room indefinitely.

Commands for which there is a distinct notion of completion—in linguistic terminology, commands which contain a verb of perfective aspect—typically generate persistent memories. For example, Go to the hallway is interpreted as visit—go to at least once—the hallway: alo(hallway). In this case, the robot's goal is to have a "memory" of having been in the hallway; this memory proposition is set by entering the hallway, after which the memory persists indefinitely and the goal is trivially satisfied from then on.

A challenge in creating a correct mapping is that the negation of a command does not necessarily imply its logical negation. For example, Don't go to the hallway is most concisely expressed as the safety ¬s.hallway (literally, Always, do not be in the hallway), as opposed to specifying that the robot should infinitely often achieve a goal of not having a memory of being in the hallway. In this case, the negation of a goal yields a safety. In general, negation of a sentence in natural language does not always transparently propagate to a negation of the logical statement that the positive form of the sentence would have generated.

However, for commands that create safeties, negation is much simpler. For example, If you see an intruder, activate your camera becomes (e.intruder ⇒ s.camera), and the negated form If you see an intruder, do not activate your camera becomes (e.intruder ⇒ ¬s.camera). Negative commands are always expressed as safeties, while the positive version of the same command may not be, as in the previous example of Go to the hallway. More complex examples of generation involving combinations of these macros are given in Sect. 5.

Support for new commands can be added by first verifying the presence of the verb and the intended argument structure in VerbNet, adding the verb and information about the arguments it takes if needed by editing an XML database. For example, a bomb-defusing robot will need to understand the verb defuse, which is not contained in VerbNet. Many new commands to be added can easily be expressed using the macros for goal or alo or mapping directly to an actuator on the robot. These commands can trivially be supported by the system by marking that sense as a simple action in the LTL generation subsystem and giving the macro it is mapped to. For example, for the user study reported in Sect. 3.4, we added a defuse sense to VerbNet and mapped it to a simulated actuator of the same name.

3.3.4 Generation tree

A novel aspect of the LTL generation process is that the transformations undertaken are automatically recorded in a generation tree to allow for a more interpretable analysis of the specification generated. As shown in Fig. 3, the generation tree allows for a hierarchical explanation of how LTL formulas are generated from natural language. There is a tree corresponding to each natural language statement, rooted at the natural language statement and with LTL formulas as leaves. The intermediate nodes are automatically created by the LTL generation process to explain how the statement was subdivided and why each LTL formula was generated.

In addition to allowing the user to inspect the generated LTL, the generation tree enables mapping between LTL formulas and natural language for specification analysis. As is shown in the following sections, this allows for natural language explanations of problems detected in the specification. During execution of the generated controller (either in simulation or with a real robot), it also allows the system to answer the question What are you doing? by responding with language from the generation tree. For example, in Fig. 3, if the current goal being pursued during execution is s.mem visit lounge, the system responds: I'm currently trying to 'visit lounge'. In cases where the original instruction involves quantification, identification of the sub-goal is particularly useful. If the user enters Go to all patient rooms, the generation tree will contain a sub-tree for each patient room, allowing for clear identification of which room is relevant to any problems with the specification.

3.3.5 Controller synthesis and execution

Given an LTL formula representing a task specification and a description of the workspace topology, the efficient synthesis algorithm introduced by Piterman et al (2006) is used to construct an implementing automaton (if one exists). In combination with lower-level continuous controllers, this automaton is then used to form a hybrid controller that can be deployed on physical robots or in simulation. To obtain this hybrid controller, a transition between two discrete states is achieved by the activation of one or more low-level continuous controllers (Kress-Gazit et al, 2009; Finucane et al, 2010).

3.4 Evaluation

3.4.1 Design

To evaluate the performance of this system when used by inexperienced users, we embedded SLURP as a robot agent in a first-person-perspective 3D video game, simulating a search and rescue scenario. The participant played the role of an operator instructing a robot through natural language commands. A screenshot of the game during an interaction with the robot is given in Fig. 4.

To succeed in the scenario, the operator was required to work with the robot to search fourteen rooms on a floor, using the robot to defuse any bombs discovered while the operator rescued hostages. To increase the difficulty of the task and force greater reliance on the robot for searching, the operator needed to neutralize hostage takers who are trying to escape from the floor while it is searched. The scenario was considered a failure if the operator ever entered a room with an active (not yet defused) bomb or if too many of the hostage takers escaped. To succeed, the participant needed to command the robot to perform two tasks: navigate all rooms, and defuse all bombs found.

The user study was designed to elicit natural language commands from users without giving any explicit suggestions regarding what they should say to the robot and what commands it understood. After providing informed consent, participants were instructed that the simulated robot ("Junior") was capable of understanding natural commands and advised to communicate with it naturally ("Talk to Junior like you might talk to someone who needs instructions from you"), giving direct commands but not doing anything unnatural such as removing prepositions and articles or using "Tarzan-speak." The experimenter was permitted to answer questions about the instructions, game controls, and interface during the training scenarios, but not during the testing scenario. The experimenter did not give any suggestions regarding what the user should say to the robot or intervene in the experiment other than restarting the experiment in case of software failure.

i

Users participated in four training scenarios of increasing difficulty before attempting the full scenario described above. The purpose of the training scenarios was to introduce them to the game dynamics as well as give them simple tasks to perform with natural language before trying to complete more complex ones. The training scenariosused smaller maps and gradually introduced the actions that would need to be performed in the main scenario: neutraliz- ing hostage takers, and commanding the robot to move be- tween rooms and defuse bombs. The layout used in the final scenario contained fourteen rooms connected by hallways in a ring configuration, a layout not used in any of the train- ing exercises that made neutralizing the escaping hostage takers significantly more difficult. The locations of bombs, hostages, and hostage takers were randomly generated for each user and could be discovered by navigating to the room they were located in.

3.4.2 Results

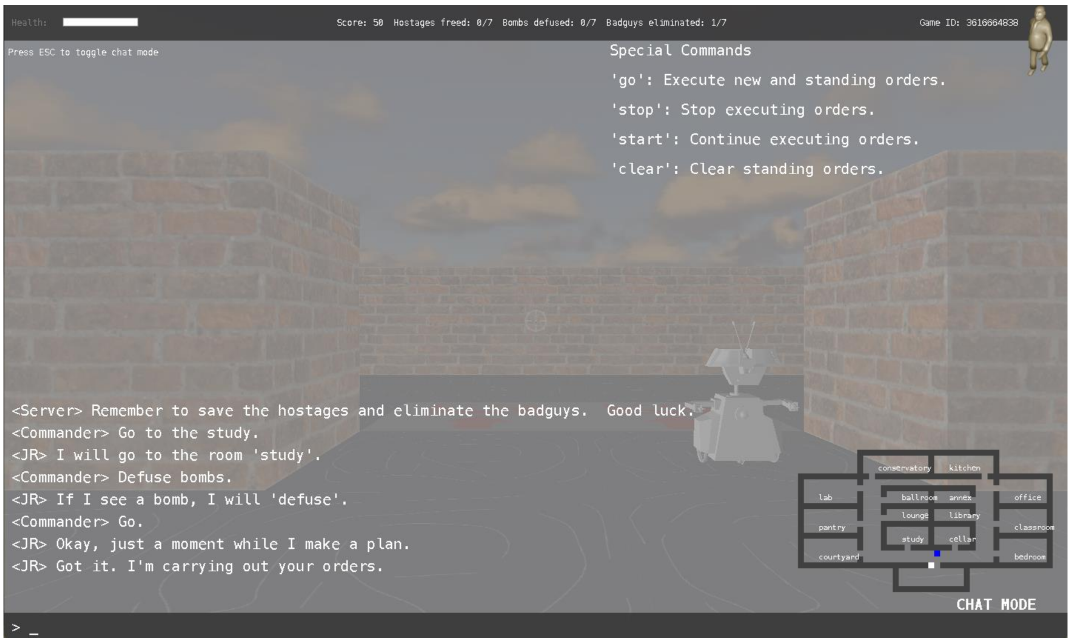
All fourteen participants successfully completed the training and testing scenarios. They were able to successfully command the robot navigate to and defuse all bombs present in the map.

We analyzed transcripts of interactions between users and the simulated robot across the four training scenarios and the final testing scenario. The transcripts consisted of 628 commands given to the simulated robot. 69 commands (11.0 %) were excluded from analysis for the following reasons: typographical errors, humorous commands unrelated to the scenario (e.g., ordering the robot to dance), non-commands (e.g., telling the robot "good job"), commands the robot cannot perform in the scenario (e.g., instructing it to move to locations not on the map), unnecessary repetition of previously not-understood commands, or if software limitations of the simulation, not the language understanding system, caused the command to not be processed. The remaining 559 commands were automatically labeled for whether they were successfully understood by the system based on its response.

526 commands (94.1 %) were understood and resulted in the successful synthesis and execution of an automaton (Table 1). The 33 commands (5.9 %) that were not understood were manually annotated and investigated to determine the cause of the failure (Table 2). The largest single cause of errors was the tagger failing to identify imperatives as verbs because they are rarely present in the training data for standard natural language processing components. For example, both Rescue the hostage and Free the hostage were not recognized as imperatives due to tagging errors.[2]

The verbs the system failed to recognize were come, destroy, find, get, and walk. The failure to recognize come (come here) and get (get up) was caused by VerbNet not including the specific imperative uses of these verbs. The

[2] Even though it was the role of the operator, not the robot, to rescue hostages, we label these examples as tagging errors because a command was given to the system and it was not properly understood. The desired response in this situation is to understand the requested action but report that the robot cannot perform it.



```
Health:                Score: 50  Hostages freed: 0/7  Bombs defused: 0/7  Badguys eliminated: 1/7        Game ID: 3616664838
Press ESC to toggle chat mode
                                                    Special Commands
                                                    'go': Execute new and standing orders.
                                                    'stop': Stop executing orders.
                                                    'start': Continue executing orders.
                                                    'clear': Clear standing orders.

<Server> Remember to save the hostages and eliminate the badguys.  Good luck.
<Commander> Go to the study.
<JR> I will go to the room 'study'.
<Commander> Defuse bombs.
<JR> If I see a bomb, I will 'defuse'.
<Commander> Go.
<JR> Okay, just a moment while I make a plan.
<JR> Got it. I'm carrying out your orders.
                                                                                    CHAT MODE
> _
```

Fig. 4: Screenshot of simulation environment

Table 1: User study performance

Overall Performance		
Result	Count	% of commands
Understood	526	94.1
Error	33	5.9

Table 2: User study error analysis

Causes of Errors			
Error Type	Count	% of commands	Example
Tagging	10	1.8	Rescue tagged as a noun in Rescue the hostages
Syntactic parsing	1	0.2	Null element not restored in Go to north rooms, then go to east rooms
Semantic parsing	2	0.4	Around not recognized as an argument in Turn around
Verb not understood	9	1.6	VerbNet sense for walk not mapped to movement action
Use of shorthand	7	1.3	One-word commands such as defuse and bedroom
Other	4	0.7	It not understood in If you see a bomb, defuse it

system was able to semantically parse sentences containing destroy and walk, but the VerbNet senses for these verbs had not been mapped to the defuse and go actions that users expected. This can be addressed by simply adding these mappings. In these cases, the user was informed that the system recognized the verb in their command but did not know what to do with it (Sorry, but I don't know how to walk), and the user discovered an alternate way to make their request (go, move), that was understood.

All but one instance of the errors due to use of shorthand consisted of saying only defuse to instruct the system to defuse bombs. Successful commands for defusing bombs produced by users included Defuse the bomb, Defuse bomb, and compound commands such as Defuse the bomb and go to the lab. Two of the errors classified as "Other" were caused by the inability of the semantic parsing system to resolve pronouns within a command: If there is a bomb, defuse it.

3.4.3 Discussion

The evaluation presented here demonstrates that inexperienced users can successfully give commands understood by SLURP without any specific knowledge of what the system is capable of understanding or the underlying lexical database or parsing system. While users did produce commands the system did not understand, they were also able to identify alternative forms that worked and complete the sce-

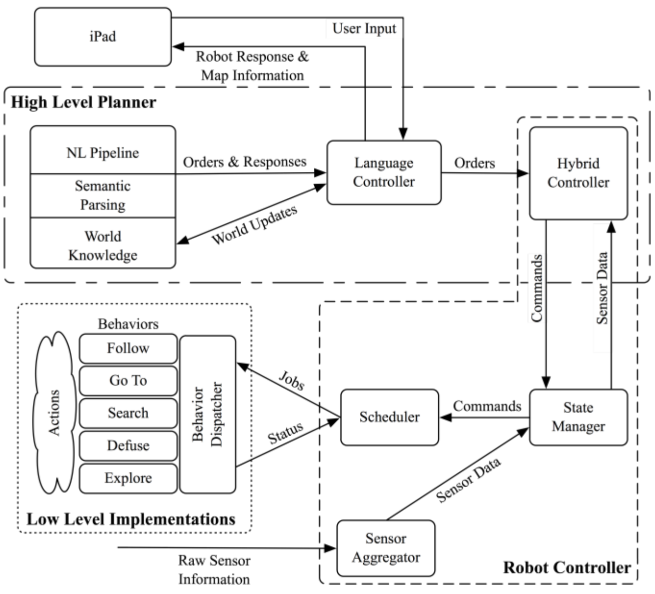
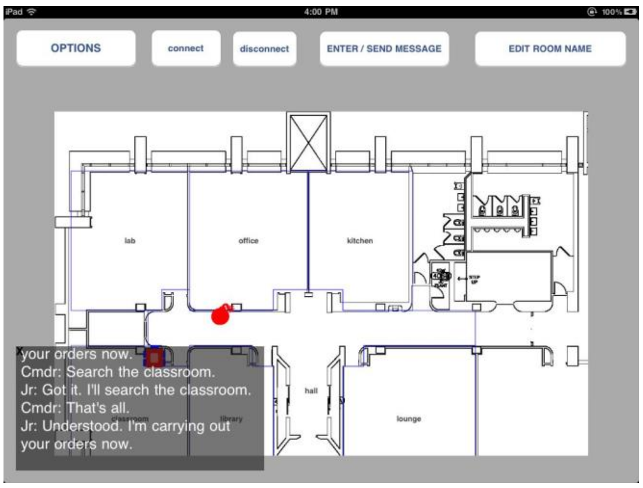


Figure 3.The end-to-end system architecture



your orders now.
Cmdr: Search the classroom.
Jr: Got it. I'll search the classroom.
Cmdr: That's all.
Jr: Understood. I'm carrying out your orders now.

Figure 4: The operator interface shows the current dialog state in the lower left along with the robot position and an icon for a bomb the robot has identified
.

narios. This was aided by the rich feedback the system provides. For example, when a user stated If there is a bomb, defuse it, the system's response was Sorry, I don't understand what you mean by 'it'. This response allows the user to identify the issue with their command and revise it. Systems that rely on a formal grammar for parsing (e.g., Dzifcak et al, 2009) report a parsing failure in this instance, not giving the user any information regarding what the issue is.

The majority of errors encountered were due to a common problem in natural language processing: differences in the type of content in the data that the tagger and parser were trained to perform well on and the actual data used in testing. The tagger and parser used are primarily trained on newswire, which contains a very small number of imperatives. This is in contrast to the data set evaluated here, where every sentence is an imperative. The data collected in this user study can be used to allow training on more relevant in-domain data, allowing for a reduction in the number of tagger and parser errors.

4 Architecture and Physical Robot Implementation

The system described above can be used with simulated robots, with graphical systems and with real robots. What differs is primarily the particulars of the low level controllers used by the resulting control automaton. For purposes of real-world experiments, we implemented a system using an iRobot ATRV Jr. The operator interacts with the system using a tablet com- puter, currently an Apple iPad. The system is comprised of modular software subsystems which were assembled into a single system using ROS (Quigley et al. 2009).

Our system's overall architecture, shown in Figure 3 is analogous in design to a three layer architecture (Gat 1998). The natural language components of the system make up the highest layer, which transforms natural language into logical statements, synthesizes a finite state automaton to carry out the requested plan, and communicates status back to the commander. The middle layer is formed by a hybrid controller, which maintains the current discrete state of the automaton given input from the environment, and the job scheduler and state manager on the robot. These modules work together to transform the logical propositions into a controller for the robot's actions. Finally, low-level continuous behaviors which primarily interact with the dynamics of the changing world are implemented on the robot to carry out the requested actions.

4.1 Operator Interface

The operator commands the system through the use of a tablet computer. Natural language utterances are the primary form of communication between the operator and robot and are entered into the system using an interface similar in design to an instant messaging or text messaging program. Information about the environment is reported by the robot using visual notifications on the display in addition to language notifications for important events, as shown in Figure 4. The interface also contains a map mode, which displays information about the layout of the world and the location of key objects within the world, including the robot's position. The map layout may be known in advance or produced by the robot as it explores. Map mode is considered to be a secondary form of communication which serves to augment the natural language interaction. It provides a source of common understanding to ground the conversation between the commander and robot by showing the objects or places relevant to the robot's operation. The map interface also displays camera imagery from the robot, allowing the commander to see various objects of interest such as bombs and hostages as they are identified by the robot.

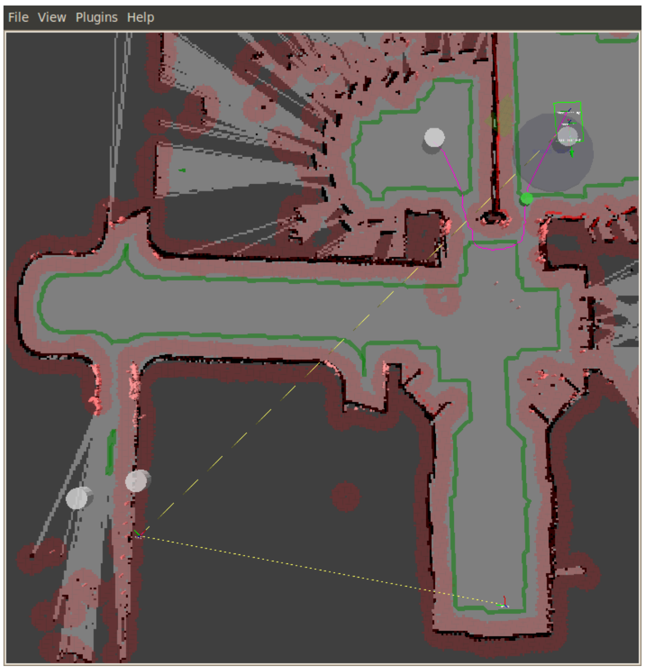


Figure 5: The robot generates a map of the environment in a fronter-based exploration scenario.

4.2 Robot Controller

The controller generated from the language deploys different low-level robot behaviors based on the goals and the state of the robot and the environment. This controller automatically reacts to different environment events, as perceived by the robot's sensors.

State and Sensor Management. One of the responsibilities of the robot is presenting information about the world as perceived by its sensors to the generated controller. The granularity of the raw sensor data is too fine for the controller, thus the data is abstracted into discrete events. Sensor output is filtered and fused with other data to create a more concise description of the world. This is done by taking pieces of raw data and interpreting them into various types of information such as abstract location, for example which room the robot is in, and what agents and objects are present in the current room. The interpreted sensor data, along with information about the location of the robot and the current behaviors being executed, form the system's state, which is reported to the controller.

Low Level Implementation. The robot is currently capable of six behaviors, including driving to a location, exploring (map building), performing a generic search of an area, following a person, retrieving (asking for) objects, and (simulated) disarming explosives. A behavior is composed of a set of rules for starting, stopping, and suspending execution, along with logic that controls the execution of one or more actions. An action is an activity, atomic at the planner's level, that the robot can perform. There are currently four actions implemented: drive, area sweep, explore, and follow. Actions react dynamically to the environment, can keep state, and even perform some lower level planning. However, they do not take into account the state of the over- all system or what other actions are currently running. Ac- tions often make use of shared resources such as drive train motors and can also be used by multiple behaviors.

Mapping, Region Discovery, and Exploration. An important aspect of being able to effectively communicate with the system is the ability for both the operator and robot to accurately refer to places in the world using names. This system supports this feature through the use of either static or dynamically generated maps. In some cases, the map of the area in which the robot is operating is known in advance, through the existence of building plans or previous experience. These maps can be preloaded into the system before deployment, which gives the operator the advantage of being able to refer to places in their instructions. However, in many cases the layout of the world is not known ahead of time, and a map must be generated. This can be achieved through the use of a strategy for frontier-based exploration (Yamauchi 1997) and simultaneous localization and mapping (Grisetti, Stachniss, and Burgard 2007). We analyze the structure of the world by drawing polygon outlines on the map to denote regions, which are assigned names. In addition, we map which polygons can be reached from other polygons using a connected graph. The polygon definitions, names, and connected graph is stored as a topological graphwhich can be used by the system for planning. We created an automated system to identify regions using an algorithm (Fabrizi and Saffiotti 2000) which uses a process of dilation and erosion of walls to determine distinct places within the world, and a process called water-shedding to determine their connectivity to each other. An example of such regions being identified as the robot explores is shown in Figure 5.

5 Examples of System Usage

We now demonstrate the system in a scenario where a person acting as the commander used a real robot to simulate a search and rescue scenario. Before retrieving the hostages, the commander needed to search the building for bombs. Hostages, bombs, and users were represented as boxes with fiducial markers in the scenario to provide a simple simulation of perception.

5.1 Known Map

The interaction shown in Figure 6 demonstrates how the commander specified a plan to the robot and received feedback as the robot understood each command, with corresponding line numbers. The commander first issued a standing order for the robot to notify the commander when any hostages were seen (line 01). When the robot was told to look for a particular human collaborator, the robot interpreted the commands as requiring a search of all rooms in which the user could be present. The request to get the defuser from the user and bring it back here shows how the system can give information about how it resolved commands to the commander. When the commander says “here,” the robot resolved it to its current location, the hall. This in ference is made explicit so the commander has the ability to correct any misunderstandings. When the commander completed giving orders (line 07), the robot formed a plan and began to execute it. The robot did not inform the commander of every action taken, in- stead only notifying when it was explicitly asked to (i.e., hostages), if it acted on a standing order, or when it completed its mission. When the robot identified the user and requested the defuser, the commander was notified. After completing the mission, the robot informed the commander that it was idle.

Not every interaction results in successful understanding. Once the robot had the bomb defuser, the commander needed to instruct the robot to use it to defuse bombs. In cases where the system was able to extract nothing of use from the utterance, in this case because VerbNet did not contain an appropriate form of disarm, the system reported that it did not understand the utterance at all (line 12). Another possible failure mode is that the system extracted the semantic structure but it did not understand how to carry out the command. In the case of the second command tried, defuse the dynamite, the system recognized that the verb was defuse and the object was the dynamite but it did not understand that dynamite is the same thing as a bomb. The responses the system gives help to make it apparent when the commander’s word choice is at fault; the system is able to express that it understood the syntax but does not know the object in this example. With such feedback, the operator may adjust its vocabulary when repeating the command (line 15).

The robot’s standing orders remain in effect across plans so that the commander does not need to specify them again for each set of commands. The standing order of informing the commander when a hostage is seen was still valid and was carried over into a new plan, as shown by the notification upon seeing a hostage (line 21). In addition, a new standing order to defuse bombs was given now that the robot has the bomb defuser. As the robot searched the lab and recognized the hostage, it notified the commander and then waited for new orders after its search was complete. The commander provided new orders to send it to the office (line 23), where it performed a search, pausing to defuse a bomb, and then completed its search of the room. The ability to remain reactive to the environment while performing actions is an important feature of the planner in a system that operates in dangerous environments where immediate reaction is critical.

In Figure 7, we give examples of how the commander was able to interrupt plans currently being executed and change the robot’s plan. The robot can be told to stop in mid-plan execution (line 05), in which case the plan is discarded. If orders have been given but the commander changes his or her mind before telling the robot to execute the commands, the commander may clear the list of orders by giving a command such as *nevermind* (line 09)

.

01 Commander: Tell me if you see any hostages.
02 Robot: I’ll let you know if I see a hostage.
03 C: Look for User 2 in the library and the classroom.
04 R: Got it. I’ll search the library and search the classroom.
05 C: Get the defuser from User 2 and come back here.
06 R: Got it. I’ll retrieve the defuser from user 2 and go to the hall.
07 C: Make it so.
08 R:Understood. I’m carrying out your orders now.
The robot searches the classroom and does not find the user. It then searches the library, identifies the user, and begins to get the defuser.
09 R: I’m now going to get the defuser in the library.
In addition to the language notification, an icon showing the location of the user is shown on the operator interface along with a picture of the user captured by the robot. After getting the defuser, the robot returns to the hall.
10 R: I’m done, and I’m in the hall.
11 C: Disarm the dynamite.
12 R: I’m sorry, I didn’t understand that at all.
13 C: Defuse the dynamite.
14 R: Sorry, I don’t know anything about dynamite.
15 C: Defuse the bombs.
16 R: Got it. I’ll defuse all bombs.
17 C: Search the lab.
18 R: Got it. I’ll search the lab.
19 C: Make it so.
20 R: Understood. I’m carrying out your orders now.
The robot drives to the lab and searches it, finding a hostage but no bombs. The robot displays a picture of the hostage on the operator’s interface.
21 R: I see a hostage.
22 R: I’m done, and I’m in the lab.
23 C: Search the office.
The robot drives to the office and searches it, finding a bomb during the search. It displays a picture of the bomb on the operator’s interface.
24 R: I’m now going to defuse in the office.
The robot defuses the bomb and then completes its search of the room.
25 R: I’m done, and I’m in the office.

Figure 6: Interaction with the robot before and during plan execution.

01 Commander: Go to the classroom.
02 Robot: Got it. I’ll go to the classroom.
03 C: Make it so.
04 R: Understood. I’m carrying out your orders now.
The robot begins driving to the lab.
05 C: Stop.
06 R: Aye sir, full stop. I’m in the hall.
The robot stops immediately.
07 C: Go to the lab.
08 R: Got it. I’ll go to the lab.
09 C: Nevermind
10 R: Okay, I’ll belay those orders.
The robot clears the “go to the lab” command from its orders.
11 C: Go to the lounge.
12 R: Got it. I’ll go to the lounge.
13 C: Make it so.
14 R: Understood. I’m carrying out your orders now.
The robot begins driving to the lounge.

Figure 7: Controlling the robot’s plan execution.

6 Conclusion

This report presents an integrated system that allows non-expert users to control robots performing high-level, reactive tasks using a natural language interface. The depth of integration between the natural language components and the synthesis, unsynthesizable core-finding, and execution modules allows for natural language specifications, feedback on specification errors, and explanation of current goals during execution.

The described user study demonstrates that the approach taken for natural language understanding, combining standard parsing and tagging modules with a deterministic semantic parsing system, is capable of understanding non-expert user commands with high accuracy. As shown in the error analysis, the greatest improvements to the performance of the system would come from training the tagger on more data containing imperatives and expanding the vocabulary coverage of the system to cover more of the verbs used by users. The design of the system makes adding support for additional verbs straightforward as there is no grammar to update, only a vocabulary list to amend.

While the use of domain-general language processing tools allows for an extensible system, some constructs common in the robotics domain may not be understood correctly. For example, a user might want to use the phrase turn on your camera instead of activate your camera as in the examples in this paper. Unfortunately, particles such as on are often assigned incorrect part-of-speech tags, resulting in failure to understand commands such as turn on and turn off. The collection of a corpus of robot control interactions for use in training broad-domain language models for robot control would result in higher performance from the natural language processing and semantic extraction components.

There are significant challenges in designing a robust mapping from natural language semantics to LTL formulas. While simple motion commands and actuations can be straightforwardly mapped into logical form, more complex actions like the delivery of objects can be more difficult to translate. However, the benefit of the proposed system is that the effort is invested just once in the design of the mapping, not repeatedly for each specification as it would be for users writing specifications in LTL or structured language. The system presented is easily applied to scenarios where the specification is largely centered around motion and simple actuations, but can be extended by users proficient in LTL to more complex scenarios by adding support for additional behaviors. Future work might explore the automatic learning of mappings between semantic structures and LTL representations. Learning a mapping that allows for complex specifications to be reliably synthesized would require the system to learn many of the subtleties of LTL specification authoring, such as maintaining consistent tense across the mapping for each type of action.

A number of issues in the LTL generation process merit further consideration. While this paper discusses some of the challenges regarding the application of negation, further work should address more formal paradigms for mapping actions to LTL formulas. The production of an ontology of common actions and the type of formulas that they produce—for example, safety conditions, adding goals, constraining the initial state—in their negated and positive forms would be a step toward a more general solution to the problem of mapping natural language to LTL. Previous work has relied heavily on grammar formalisms to ease semantic extraction. While those formalisms provide a structure for easy extraction of semantic roles, they are not robust to natural input and do not address the more urgent problem of robust logical representations over large sets of possible actions which remain appropriately synthesizable in complicated specifications. The examples presented here have focused on the execution of a single specification. However it is possible that over the course of a mission requirements may change and new commands may be given. While the underlying execution environment supports resynthesis and transitioning execution to a new automaton during execution, this introduces a number of challenges in the communication between system and user. Future work should explore means for maintaining the level of natural language integration presented in this paper across more complex execution paradigms, handling events such as changes in the workspace topology, as would occur when operating in environments that are only partially known.

.

References

Berant J, Liang P (2014) Semantic parsing via paraphrasing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1415–1425

Bhatia A, Kavraki LE, Vardi MY (2010) Sampling-based motion planning with temporal goals. In: IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp 2689–2696

Biere A (2008) PicoSAT essentials. Journal on Satisfiability, Boolean Modeling and Computation (JSAT) 4:75–97

Bikel DM (2004) Intricacies of Collins' parsing model. Computational Linguistics 30(4):479–511

Bobadilla L, Sanchez O, Czarnowski J, Gossman K, LaValle S (2011) Controlling wild bodies using linear temporal logic. In: Robotics: Science and Systems (RSS)

Brooks D, Lignos C, Finucane C, Medvedev M, Perera I, Raman V, Kress-Gazit H, Marcus M, Yanco H (2012) Make it so: Continuous, flexible natural language interaction with an autonomous robot. In: Proceedings of the Grounding Language for Physical Systems Workshop at the Twenty-Sixth AAAI Conference on Artificial Intelligence

Chen DL, Mooney RJ (2011) Learning to interpret natural language navigation instructions from observations. In: Proceedings of the Twenty-Fifth AAAI Conference on Artifical Intelligence, pp 859–865

Cizelj I, Belta C (2013) Negotiating the probabilistic satisfaction of temporal logic motion specifications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4320–4325

Clarke EM, Grumberg O, Peled DA (1999) Model Checking. MIT Press

Dzifcak J, Scheutz M, Baral C, Schermerhorn P (2009) 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: IEEE International Conference on Robotics and Automation (ICRA), pp 4163–4168

Fabrizi, E., and Saffiotti, A. 2000. Extracting topology-based maps from gridmaps. Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on 3:2972–2978 vol. 3.

Fainekos GE (2011) Revising temporal logic specifications for motion planning. In: IEEE International Conference on Robotics and Automation (ICRA), pp 40–45

Finucane C, Jing G, Kress-Gazit H (2010) LTLMoP: Experimenting with language, temporal logic and robot control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 1988–1993

Gabbard R, Marcus M, Kulick S (2006) Fully parsing the Penn Treebank. In: Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (NAACL HLT), pp 184–191

Gat, E. 1998. Three-Layered Architectures. AI-based Mobile Robots: Case Studies of Successful Robot Systems 195–210. Chapter 8.

Grisetti, G.; Stachniss, C.; and Burgard, W. 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. Robotics, IEEE Transactions on 23(1):34–46.

Karaman, Frazzoli (2009) Sampling-based motion planning with deterministic μ-calculus specifications. In: IEEE Conference on Decision and Control (CDC), pp 2222–2229

Kim K, Fainekos GE, Sankaranarayanan S (2012) On the revision problem of specification automata. In: IEEE International Conference on Robotics and Automation (ICRA), pp 5171–5176

Kloetzer M, Belta C (2008) A fully automated framework for control of linear systems from temporal logic specifications. IEEE Transactions on Automatic Control 53(1):287–297

Kress-Gazit H, Fainekos GE, Pappas GJ (2008) Translating structured english to robot controllers. Advanced Robotics 22(12):1343–1359

Kress-Gazit H, Fainekos GE, Pappas GJ (2009) Temporal-logic-based reactive mission and motion planning. IEEE Transactions on Robotics 25(6):1370–1381

Matuszek C, Fox D, Koscher K (2010) Following directions using statistical machine translation. In: Human-Robot Interaction (HRI), pp 251–258

Matuszek C, FitzGerald N, Zettlemoyer L, Bo L, Fox D (2012) A joint model of language and perception for grounded attribute learning. In: Proceedings of the 29th International Conference on Machine Learning (ICML), pp 1671–1678

Matuszek C, Herbst E, Zettlemoyer L, Fox D (2013) Learning to parse natural language commands to a robot control system. Experimental Robotics 88:403–415

Piterman N, Pnueli A, Sa'ar Y (2006) Synthesis of reactive(1) designs. In: Verification, Model Checking, and Abstract Interpretation (VMCAI), pp 364–380

Poon H, Domingos P (2009) Unsupervised semantic parsing. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1–10

Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; and Ng, A. 2009. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software.

Raman V, Kress-Gazit H (2011) Analyzing unsynthesizable specifications for high-level robot behavior using LTLMoP. In: Computer Aided Verification (CAV), pp 663–668

Raman V, Kress-Gazit H (2013a) Explaining impossible high-level robot behaviors. IEEE Transactions on Robotics 29:94–104

Raman V, Kress-Gazit H (2013b) Towards minimal explanations of unsynthesizability for high-level robot behaviors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 757–762

Raman V, Kress-Gazit H (2014) Unsynthesizable cores—Minimal explanations for unsynthesizable high-level robot behaviors. arXiv:1409.1455

Raman V, Lignos C, Finucane C, Lee KCT, Marcus M, Kress-Gazit H (2013) Sorry Dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language. In: Robotics: Science and Systems (RSS)

Schuler K (2005) Verbnet: A broad-coverage, comprehensive verb lexicon. PhD thesis, University of Pennsylvania

Tellex S, Kollar T, Dickerson S, Walter MR, Banerjee AG, Teller SJ, Roy N (2011) Understanding natural language commands for robotic navigation and mobile manipulation. In: Proceedings of the Twenty-Fifth AAAI Conference on Artifical Intelligence, pp 1507–1514

Toutanova K, Klein D, Manning CD, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technol- ogy (NAACL HLT) - Volume 1, pp 173–180

Wongpiromsarn T, Topcu U, Murray RM (2010) Receding horizon control for temporal logic specifications. In: Hybrid Systems: Computation and Control (HSCC), pp 101–110

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on 146–151.