RoboFrameNet: Verb-Centric Semantics for Actions in Robot Middleware

Brian J Thomas and Odest Chadwicke Jenkins

***Abstract*— Advancements in robotics have led to an ever-growing repertoire of software capabilities (e.g., recognition, mapping, and object manipulation). However, robotic capabilities grow, the complexity of operating and interacting with such robots increases (such as through speech, gesture, scripting, or programming). Language-based communication can offer users the ability to work with physically and computationally complex robots without diminishing the robot's inherent capability. However, it remains an open question how to build a common ground between natural language and goal-directed robot actions, particularly in a way that scales with the growth of robot capabilities. We examine using semantic frames – a linguistics concept which describes scenes being acted out – as a conceptual stepping stone between natural language and robot action. We examine the scalability of this solution through the development of RoboFrameNet, a generic language-to-action pipeline for ROS (the Robot Operating System) that abstracts verbs and their dependents into semantic frames, then grounds these frames into actions. We demonstrate the framework through experiments with the PR2 and Turtlebot robot platforms and consider the future scalability of the approach.**

I. Introduction

Robotics systems have become increasingly complex in terms of functionality and operating difficulty. While the first property is desirable in that it increases the robot's capabilities to perform required tasks, the second is much less so, as it places additional burden on the robot's operator. An interaction schema which manages operational complexity while allowing for functionally complex robots is highly desirable.

Natural language interaction provides an intuitive interface while enabling the specification of complex functionality. The scope of natural language commands is the scope of the language itself, which describes a multitude of real-world actions and events. Nearly every person on Earth speaks some form of natural language. When people perform services for each other, the request is very often written or spoken.

Semantic frames (Ruppenhofer et al. [7]) enable natural language to be an operational medium for robotics by acting as a stepping stone in the gap between natural language and goal-directed robot actions. Semantic frames, as elaborated in further sections, describe a scene being acted out; this concept is shared in natural language (as scene understanding) and in robot action (as a plan or goal for action). Thus, the gap between natural language and robot action can be traversed by abstracting from language to semantic frames, then grounding semantic frames into goal-oriented robot actions.

Fig. 1. Instructing a Turtlebot to follow the operator.

The **RoboFrameNet**[1] framework describes a system that uses natural language to command robot action through the intermediary of semantic frames. Below, we detail the conceptual approach to RoboFrameNet, describe an implementation of the RoboFrameNet framework, and demonstrate the framework's ability to ground natural language commands into robot actions through implementing a selection of tasks within the framework.

II. Background and Related Work

Natural language understanding's beginnings can be traced back to work by Winograd [9] on SHRDLU, a system which processed natural language instructions and performed actions in a virtual environment. Inspired by this system, researchers pushed forward trying to extend SHRDLU's capabilities into real-world environments and soon branched into tackling various subproblems, including NLP (natural language processing) and robotics systems.

In NLP, several domain-specific systems have been created; for instance, Kollar et al. [4] and MacMahon et al. [5] have developed methods of following route instructions given in natural language. Dzifcak et al. [2] studied translating natural language instructions into goal descriptions and actions. Chernova et al. [1] implemented natural language and action-oriented human-robot interaction with humans in a task by data-mining previous human-human interactions

This work was supported in part by NSF CAREER Award IIS-0844486 and by ONR PECASE Award N000140810910.

O.C. Jenkins is a professor in the Department of Computer Science at Brown University, Providence, RI 02912, USA. cjenkins@cs.brown.edu

Brian Thomas is a graduate student in the Department of Computer Science at Brown University, Providence, RI 02912, USA. brian@cs.brown.edu

[1] http://www.ros.org/wiki/roboframenet

of the same task. However, the scalability of these solutions outside of their test domain remains open.

In robotics systems, conducted research has produced frameworks for robot middleware such as ROS, developed by Quigley et al. [6], which has been used in several domains of modern robotics research.

Attempts have been made to recombine the fields of NLP and robotics systems. Building on ROS, Tenorth et al. [8] has developed robotic systems capable of inferring and acting upon implicit commands using knowledge databases.

Outside of computer science, linguists have examined problems related to understanding verbs; for instance, FrameNet (Ruppenhofer et al. [7]) portrays verbs as a key role of a "scene" or semantic frame.

III. Approach

We approach the problem of grounding natural language into robot action by recognizing a concept shared by the domains of natural language and robot actions, namely that of **semantic frames**. A semantic frame encapsulates the concept of a scene being acted out, complete with actors, objects, and other necessary descriptors. In natural language, a semantic frame can be interpreted as the result of scene understanding. In robotics, a semantic frame can be thought of as a plan for robot action. The term "semantic frame", along with several others described in the next section, have been borrowed from the linguistics community – particularly work by Ruppenhofer et al. [7] on FrameNet – and will be applied to describe RoboFrameNet's analogous features. A brief description of FrameNet is included as the next section to introduce relevant terms.

As illustrated in Figure 2, establishing semantic frames as a stepping stone between language and action allows the problem to be broken into two separate tasks: first, the abstraction from natural language to semantic frames; and second, the grounding of semantic frames into robot action. To accomplish these tasks, several smaller steps within each task will be detailed. Simultaneously, an implementation following the to-be-outlined traversal will be discussed.

For the purpose of implementation, a few high-level choices must be considered. First, ROS was chosen to implement this approach because of its community support for several robotics platforms and its modularized codebase. Programs developed for ROS are separated into constructs called nodes, which interact with each other via inter-process communication. Nodes perform functions such as navigating from one location to another, processing an image, providing an interface to a software library, or abstracting a specific piece of hardware.

Second, a subset of nodes called *action servers* was chosen to serve the role of goal-directed robot actions. Action servers perform preemptible high-level actions upon request; for instance, navigation, object manipulation, and arm motion tasks may be implemented as action servers. Since the robotic system needs to perform tasks on demand, action servers fill the roll of robot actions fittingly.

The support of action servers eases programming burden, and ROS's hardware abstraction allows this work to function on several robots. However, this help does not solve the fundamental grounding problem.

IV. FrameNet

We describe FrameNet and its terminology in this section. FrameNet uses several terms not generally seen outside of the linguistics community; words relevant to parts of RoboFrameNet are described below as they apply to the original FrameNet. The RoboFrameNet analogs of FrameNet concepts will be described in further sections as appropriate.

As mentioned above, a **semantic frame** describes a scene being acted out, complete with actors (active members of the scene) and objects (passive members of the scene). Additionally, semantic frames possess a certain level of generality, but not an excessive one. This level is best described via example. A semantic frame describes a scene on the level of generality of "an *actor* giving a *recipient* an *object*". This is more general than describing "a *robot* giving *Roger* a *sandwich*". However, it is less general than describing "an *actor*, *recipient*, and *object* interact". Empirically, in our work, this level of generality is reasonable for both the abstraction and grounding tasks. For natural language, abstraction to a semantic frame is similar to scene understanding; for grounding robot actions, grounding from a semantic frame is similar to running a program with a set of provided arguments.

A semantic frame is said to be *evoked* by a particular word. In its simplest case, this word is a verb. However, as the work of Ruppenhofer et al. [7] demonstrates, a verb alone does little to describe the scene. For example, the full meaning of verb "give" in a sentence cannot be understood without knowing about the words related to it, such as who the giver and recipient are and what the object being given is. These related words are termed **frame elements**; more explicitly, they are defined as items that are both related to a frame-evoking word and relevant to the frame evoked. (Note that the word which evokes a semantic frame is not typically a frame element.) Each frame element contains a description of the role of the element ("recipient") as well as, when filled, the object playing that role ("Roger"). Additionally, frame elements contain a concept known as **coreness** – that is, whether a given frame element is critical (in other words, core) to the scene. For example, the recipient would be considered core, but the method of giving ("gently", "forcefully", "reluctantly"), while adding detail to the frame, is not necessary to generally comprehend the frame and thus would not be core.

Semantic frames and the frame elements within therefore describe scenes being acted out. One may ask, then, how does natural language get mapped into semantic frames? This job is the role of the **lexical unit**. In FrameNet, lexical units are hand-annotated mappings of natural language sentences. These mappings highlight each frame element's relevant words in each sentence or utterance. Presumedly, the goal of this hand-annotation is to provide a corpus upon which

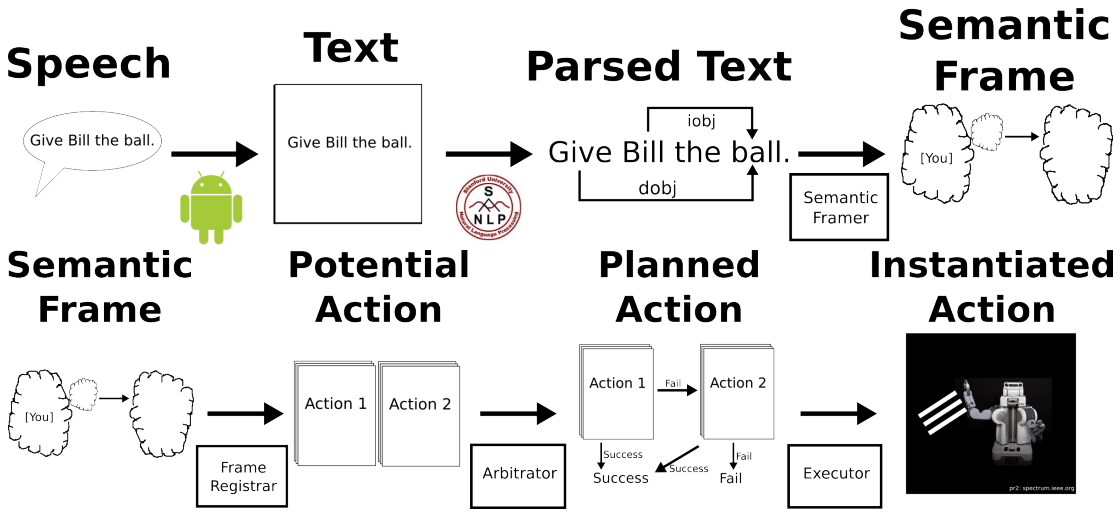

Fig. 2. The RoboFrameNet processing pipeline, as explained in Sections V and VI. The first line (Section V) abstracts spoken or written natural language into semantic frames. The second (Section VI instantiates goal-directed robot actions from semantic frames.

humans or machines can learn semantic frames and their mappings, enabling them to then create their own mappings.

These terms – semantic frames, frame elements, coreness, and lexical units – are recycled below to describe analogous features within RoboFrameNet.

V. Natural Language to Semantic Frames

A. Concept

The goal of the first half of the language-to-action pipeline is to process natural language input into semantic frames. This task will be accomplished by parsing the input sentences using a dependency parser, then mapping the output dependencies to frame elements in one or more of RoboFrameNet's semantic frames.

Natural language enters the RoboFrameNet system as either voice or text input. Speech can be (and is) converted into text using a speech-to-text engine. Text from speech may be confirmed by the speaker. Thus, without loss of generality, the input to the system is established as natural language text.

The key insight to processing natural language into semantic frames is that regardless of the actual words used, their grammatical relation to other words in the sentence is always the same. For instance, the direct object of the word "give" is the object to be given, and the indirect object is the recipient, regardless of who the recipient or what the object is. Given these grammatical relations (direct object, indirect object, etc.), mappings can be formed from each verb's related words – and the relation's type – to each relevant semantic frame and its frame elements. In RoboFrameNet, this mapping is specified using **lexical units**. A lexical unit contains a verb, the verb's corresponding semantic frame, and a set of mappings from grammatical relations to frame elements. For instance, the system may have a lexical unit with the verb *give*, the corresponding semantic frame *give_object*, and the mappings *indirect object -¿ recipient* and *direct object -¿ object_to_give*. (The semantic frame *give_object* would then contain the frame elements *recipient* and *object_to_give*.)

For a lexical unit to form a valid mapping, it must fill in all elements that are specified to be **core** in the corresponding semantic frame. Non-core elements may optionally be filled in. Thus, a set of grammatical relations may form a semantic frame through the appropriate lexical unit.

Now, the only question that remains is how to obtain grammatical relations from natural language. Fortunately, this problem is well-studied in natural language (for instance, by Klein and Manning [3]) and is known as dependency parsing.

B. Implementation

To obtain semantic frames from natural language, two existing technologies are leveraged for language processing by wrapping each in the ROS framework. First, if speech input is given, it is converted into text using Android[2] voice recognition services (*voice_recognition*[3]). Next, text is sent through the Stanford dependency parser (Klein and Manning [3]; wrapped in ROS as *stanford_parser_ros*[4]). From the dependency parser, a parse tree and a set of dependencies are output. After this, the sentence's verb is determined using the parse tree, and all lexical units specifying this verb are recalled (*semantic_framer*[5]). Then, an attempt is made to fill each lexical unit using the dependencies. Successfully filled lexical units are passed to the *frame_registrar*[6]. (Filled lexical units trivially and directly fill semantic frames.)

Lexical units are specified using YAML (Yet Another Markup Language) files. As described in the conceptual section above, a lexical unit consists of a verb, its related semantic frame, and a set of mappings from grammatical

[2] http://developer.android.com
[3] http://www.ros.org/wiki/voice_recognition (Available soon.)
[4] http://www.ros.org/wiki/stanford_parser_ros
[5] http://www.ros.org/wiki/semantic_framer
[6] http://www.ros.org/wiki/frame_registrar

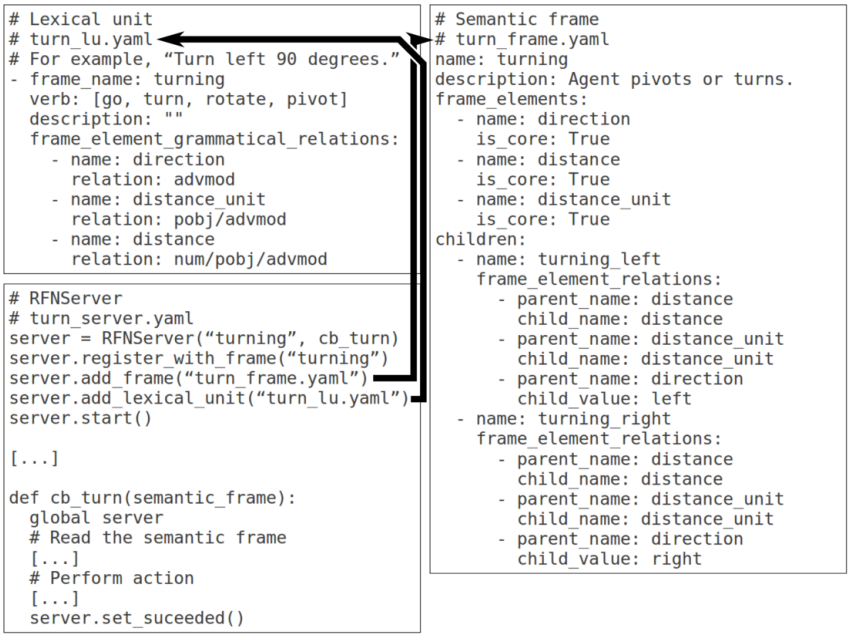

Fig. 3. Clockwise from top left: A lexical unit, semantic frame, and RFNServer used in RoboFrameNet. Files such as these are used to describe and add new capabilities to RoboFrameNet. The semantic frame describes a new scene. The lexical unit uses natural language dependencies to fill this semantic frame, and the RFNServer accepts the filled semantic frame in order to perform an action on the robot. The collection above adds the capability of turning the robot, allowing sentences such as "Turn right 2 radians." and "Turn left 120 degrees." to be commanded and executed. Note also that the RFNServer is dynamically adding relevant semantic frames and lexical units to the RoboFrameNet pipeline at runtime.

relations to frame elements. The contents of a lexical unit are directly related to the YAML file (see figure 3).

Finally, we describe the relation between FrameNet's implementation of lexical units with RoboFrameNet's implementation of the same. In FrameNet, lexical units are example sentences with hand-annotated frame elements. They are intended to provide sample mappings of natural language to semantic frames. In RoboFrameNet, lexical units are intended to provide general, automatic mappings from natural language to semantic frames. Instead of hand-annotating frame elements, lexical units automatically determine frame elements by using a sentence's grammatical relationships. Thus, while the goal of lexical units in each case differs (as an exemplar in FrameNet or as a general mapping in RoboFrameNet), their function is equivalent – to map natural language sentences to semantic frames.

VI. Semantic Frames to Robot Action

At the end of the first half of the processing pipeline, a number of filled semantic frames are received. The goal of the second half of the pipeline, then, is to take these frames and map them to robot actions. The overarching idea used is that programs will register their ability to perform a scene with the corresponding semantic frame, so that a filled semantic frame can request that the registered program run, given the filled frame elements.

Like in FrameNet, semantic frames are not isolated elements. Rather, they are connected in a directed acyclic graph. From the received filled semantic frames, the tree with the filled semantic frame as the root is traversed in search of other semantic frames that are equally capable of describing the task at hand – along with their associated registered callbacks. This process outputs a set of potential robot actions.

Given a set of potential robot actions, a plan is formed in the shape of a state machine. This plan can be formed using any number of existing planning algorithms; for instance, one could determine the best program candidates from prior experience using machine learning. More naively, one could pick one action at random.

Finally, this state machine is executed and thus physically instantiated on a robot.

A. Implementation

The *frame_registrar* receives the filled lexical units from the *semantic_framer* (as described in Section V-B) and fills out the corresponding semantic frames.

Semantic frames are specified using YAML files. In RoboFrameNet, a semantic frame consists of its name, a description of the scene being acted out, its frame elements, and their coreness to the scene. Further, semantic frames may contain mappings to their children, enabling the conceptually-described graph structure. In this case, mappings between related semantic frames' frame elements are provided either by specifying the corresponding frame elements or by requiring one or more frame element's argument to satisfy one of a list of values. The contents of a semantic frame are directly related to the YAML file (see figure 3). (Note: FrameNet's semantic frames contain the same components – with the same existential purpose

– as those in RoboFrameNet, with the minor distinction that in FrameNet, inter-frame relations are described in a separate file instead of with the parent frame, as is done in RoboFrameNet.)

Registering a callback to a semantic frame is performed using a library call from an RFNServer. (See section VII.)

Next, a search through the semantic frame graph is performed using these filled semantic frames to find other candidate frames to complete the action. In our implementation of RoboFrameNet's *frame_registrar*, two different types of relations for semantic frames were explored, namely, one relation for specificity and one for sequentiality. The first (frame specificity) allows one to associate a (child) semantic frame as able to execute a subset of the actions of another (parent) semantic frame. For instance, the semantic frame "giving_drink" may be a child of the semantic frame "giving" in this way; the objects one would give in *giving_drink* is a subset of the objects one would give in giving. The second (frame sequentiality) allows one to specify a sequence of child actions which is able to perform the same task as the parent semantic frame. For instance, "giving" may be specified as the sequence of actions "[extending_arm, ungrasping]". (These two types of relations will make a correctly-constructed graph of semantic frames directed and acyclic.)

Once the graph of semantic frames has been searched and a list of all fitting program callbacks has been determined, a state machine is formed (*arbitrator*[7]). Due to time constraints, intelligent algorithms for this section were not explored in depth; rather, matching callback sequences are randomly ordered.[8] The first series of callbacks that completes successfully terminates the state machine; upon failure, the next series begins execution. (It is implicitly – and in some cases falsely – assumed that the world state does not change due to a failed sequence.)

Finally, this state machine is executed (*executor*[9]). RFNServers (RoboFrameNet Servers) – a subclass of action servers – perform these actions when they are called. (See Section VII.) First, if an action server is running from a previous command, it is preempted. Then, the relevant sequence of action servers for the new command are called in series.

VII. RFNServers

RFNServers (RoboFrameNet Servers) allow programs to access the features of RoboFrameNet by providing library calls for registering a program callback to a semantic frame, adding new semantic frames, and adding new lexical units. A program which is an ActionServer can be trivially modified to be an RFNServer by changing the instantiation line. (RFNServers are simply a subclass of action servers.) Additionally, RFNServers require a new callback which receives a single argument – a filled semantic frame. This callback is used when the *executor* decides to run the registered RFNServer. Sample code is provided in Figure 3 to illustrate instantiation, frame registration, adding a semantic frame, adding a lexical unit, and creating a callback function.

VIII. Evaluation

We measure RoboFrameNet's success by the number of tasks and variety of domains described and grounded. These metrics exhibit RoboFrameNet's capability to meet the goal of scaling to all possible ROS functionalities. For this evaluation, we selected and created 52 semantic frames for tasks in 10 domains. For a selection of 6 domains and 16 semantic frames in this set, we abstracted natural language to these frames and grounded these frames in robot action. PR2 and Turtlebot platforms performed these tasks and demonstrated RoboFrameNet's agnosticism towards robotic platform.

RoboFrameNet's generality was demonstrated by grounding the system over 10 different domains and completing the pipeline (including natural language bindings and robot actions) in 6 of these. Domains in which the pipeline was completed were (with a representative example): following the operator ("Follow me.")[10], speaking such as emitting a mooing sound ("Speak.")[11], moving or turning a set amount based on directions ("Move forward 3 feet.")[12], system aliveness tests ("Ping.")[13], navigating on a map ("Go to the office.")[14], and giving props such as high fives ("Give me 5.")[15]. Each example given above was executed correctly on a robot – the first four on a Turtlebot and the last three on the PR2.

An additional 4 domains in which semantic frames were developed but the pipeline was not completed were (with a hypothetical representative example): cheffing ("Chop the cucumber."), cleaning the kitchen ("Scrub the pot."), laundering ("Fold the towel."), and pick and place ("Move the cup to the sink.").

IX. Strengths and Limitations

RoboFrameNet provides a domain-independent pipeline for grounding natural language into robot action while enabling domain-dependent groundings via lexical units and semantic frames. This combination allows a wide variety of tasks – including many tasks yet to be discovered – to be grounded in robot action from natural language input.

However, there are still a few issues with our implemented system. For one, in almost all cases, the Stanford dependency parser makes each provided dependency a single word. This makes it difficult to provide lexical units for sentences such as "Go to Roger's office.". Because the direct object is "office" and not "Roger's office", one needs to additionally specify the possessive of the direct object ("Roger") for this and similar sentences. This can make the task of creating

[7] http://www.ros.org/wiki/arbitrator

[8] Note that callbacks are sequences of semantic frames: recall the sequential semantic frame relationship.

[9] http://www.ros.org/wiki/executor

[10] http://www.ros.org/wiki/turtlebot_follower

[11] http://www.ros.org/wiki/moo

[12] http://www.ros.org/wiki/move_action_server

[13] http://ros.org/wiki/rfnserver (Included in package)

[14] http://www.ros.org/wiki/navigation

[15] http://www.ros.org/wiki/pr2_props

Fig. 4. Tasks performed with the PR2 and Turtlebot using RoboFrameNet. From left to right, the instructions given were: "Come to the fridge.", "Follow me.", "Give me 5."

lexical units more arduous than we believe is necessary in an ideal system. (On the other hand, the form of specification provided by the parser does makes the job of the programmer easier, as each argument of the semantic frame is more rigidly described.)

Relatedly, our current implementation will ignore words for which it is not instructed to search. Using our previous example ("Go to Roger's office."), unless a lexical unit exists which instructs the system to search for the possessive of the direct object ("Roger"), the word will be ignored, and the sentence will be interpreted equivalently to that of "Go to the office.", "Go to Buck's office.", "Go to the furthest office away.", and other variations.

Our current implementation only handles compounded words if they are explicitly coded in the lexical units. (Words such as "and" and "or" generate dependencies in the Stanford dependency parser.) However, it would be possible to automatically convert such a series of words to a list and fill the relevant frame element with this list.

Finally, on a more technical level, the scalability of ROS's action servers is limited. Because each server, when inactive, still takes up some small amount processing power, there is a limit as to how many may be running at once. Further, different software systems running on robots such as the PR2 often require different hardware settings for running. (For instance, the systems may require different camera parameters such as resolution, frame rate, zoom, and focus.) This problem could be solved by shutting down and starting up the entire driver stack between actions, but this may not be desirable or even feasible.

However, even given these limitations, RoboFrameNet as implemented can ground a broad range of instructions and commands. Further, as natural language research progresses, we believe that future improvements to those systems will integrate into and improve the first half of the RoboFrameNet pipeline.

X. Conclusion

Communication with robots through dialog allows users to interact with operationally complex robots without diminishing either the necessary expressiveness of the commands or the abilities of the robot. The proposed RoboFrameNet pipeline will facilitate this communication both by grounding verbs into concrete robot actions through the use of semantic frames and by providing programs access to a natural language interface. Because RoboFrameNet is domain agnostic, this model can scale to ROS's capabilities. ROS offers community code support and provides a framework upon which to implement this model. Experiments using the Turtlebot and PR2 platforms demonstrate the aforementioned claims and capabilities.

References

[1] Sonia Chernova, Jeff Orkin, and Cynthia Breazeal. Crowdsourcing hri through online multiplayer games. *AAAI Fall Symposium on Dialog with Robots*, pages 14–19, 2010.

[2] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul 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. *ICRA*, 2009.

[3] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[4] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. *International Conference on Human-Robot Interaction*, pages 259 – 266, 2010.

[5] Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *AAAI*, 2006.

[6] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. *Proceedings of the Open-Source Software workshop at the International Conference on Robotics and Automation (ICRA)*, 2009.

[7] Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. *FrameNet II: Extended Theory and Practice*. 2010.

[8] Moritz Tenorth, Lars Kunze, Dominik Jain, and Michael Beetz. Knowrob-map - knowledge-linked semantic object maps. *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*, 2010.

[9] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MIT, February 1971.