Verb Semantics for Robot Dialog

Brian J Thomas and Odest Chadwicke Jenkins
Department of Computer Science
Brown University
Providence, Rhode Island 02912
Email: {brian,cjenkins}@cs.brown.edu

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 in human language that will scale with the growth of robot capabilities, for instance within development environments such as ROS (the Robot Operating System). We examine this scaling problem through large-scale symbol grounding for robot dialog. We explore this problem in two parts through the development of: (1) a generic software framework for ROS that grounds parts of speech (verbs for now) into robotic capabilities using the proposed action hierarchy model and (2) a dialog interface for human-robot interaction through an expressive subset of natural language. We will evaluate the framework and interface through mobile manipulation experiments with a PR2, with consideration of the future scalability of our approach.**

I. Background and Related Work

Winograd [8] developed SHRDLU, a system which processed natural-language instructions and performed actions in a virtual environment. From this, researchers pushed forward trying to extend SHRDLU's capabilities into real-world environments and soon branched into tackling various sub-problems, including NLP and robotics systems. Research conducted on the robotics systems side has resulted in frameworks such as ROS, developed by Quigley et al. [5], which has been used in several domains of modern robotics research. NLP research including robotic components has also lead to advancements. Notably, Kollar et al. [3] and MacMahon et al. [4] 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 of the same task. However, the scalability of these solutions outside of their test domain remains open. Attempts have been made recombining these fields. For instance, Tenorth et al. [7] has developed robotic systems capable of inferring and acting upon implicit commands using knowledge databases. Finally, linguists have also examined problems related to understanding verbs; for instance, Ruppenhofer et al. [6]'s FrameNet portrays verbs as a key role of a "scene" or semantic frame.

II. Contributions and Implementation

We approach the problem using an **action hierarchy model**, which binds verbs in input dialog to actions in ROS. **Dialog** is defined an expressive subset of natural language and is a starting point for more sophisticated language processing. Here, we use dialog to establish communication patterns that both robots and humans can understand, establishing a common ground between humans and robots. As NLP continues to improve, common ground can be established in ways even more similar to natural language than dialog. Further, grounding can be established not only for verbs, but for other parts of speech such as nouns, prepositional phrases, and the like.

We choose ROS to implement this approach because of its community support for a multitude of platforms and localization packages. Further, code developed for ROS is already modularized into constructs called nodes, which interact with each other via inter-process communication. The combination of community and modularization provides us with a plethora of code chunks that verbs can ground themselves in. Because of ROS's community's size and heterogeneity, we need to also consider how the hierarchy will be stored, who will have permission to modify it, and how. Centralized control (in the form of, e.g., a "blessed" verb grounding package) offers better reliability and control of user experience than community control but at the cost of the community being able to contribute. To get the best of both worlds, we will develop both a "blessed" verb grounding package and a patch system, the former creating a positive user experience and the latter enabling the community to link their work to ours without requiring they go through a formal appeals process.

III. Action Hierarchy Model

A. Concept

The action hierarchy model is a four-level model which binds verbs in dialog to actions in ROS. Namely, the four levels (from highest to lowest) are instructions, verbs, action classes, and action instances. **Instructions** are dialog language input and can be thought of as a state machine that the robot should execute, with each verb in the dialog representing a single state. Instructions can turn into verbs by creating this state machine to structure the dialog-implicit transitions between verbs. Each **verb** is associated with a single action class, so verbs turn into action classes by simply referring

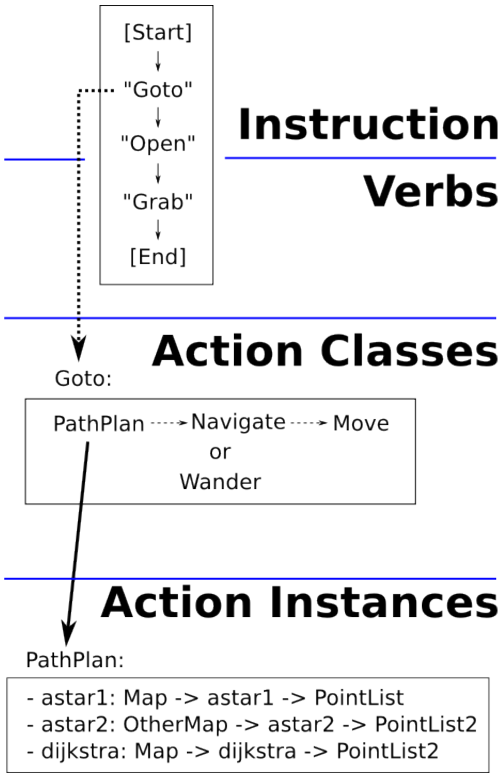
[Start]
"Goto"
"Open"
"Grab"
[End]
Instruction Verbs
Action Classes
Goto:
PathPlan
Navigate
Move
or
Wander
Action Instances
PathPlan:
- astar1: Map -> astar1 -> PointList
- astar2: OtherMap -> astar2 -> PointList2
- dijkstra: Map -> dijkstra -> PointList2

Fig. 1. An example of grounding verbs in dialog into ROS actions using the action hierarchy model.

to the given action class. An **action class** is a new construct which generalizes actions to the tasks they solve. For instance, path planning (PathPlan) is one possible action class, with the task being constructing a route. Action classes are associated with each other and action instances in a directed acyclic graph (DAG); action classes turn into action instances by the method described in the next paragraph. **Action instances**, in contrast to the abstraction of action classes, are concrete pieces of code in ROS which implement specific solutions to an action class's task. Pieces of code implementing Dijkstra's algorithm and the A* algorithm could be considered action instances of PathPlan.

Each action class consists at least one of at least one of the following: a set of one or more action classes, along with their interactions; or an action instance. PathPlan, in this example, refers only to the latter type of action class members and would contain A*, Dijkstra, etc. code blocks as its members. Two possible examples of the former type of action class member could be located within a Goto action class (referred to by the verb "go to"). One member would be the action classes PathPlan, Navigate, and Move and their interaction; as a whole, these describe deliberative route planning, navigation, and movement for going to a new location. Another member would be the action class Wander, which would describe going to a new location as a guess-and-check algorithm. Action class can turn into action instances by choosing exactly one member of each action class encountered in the DAG. (However, note that, for instance, Goto's member "PathPlan + Navigate + Move" is considered a single choice, so a single action class can result in the creation of multiple action instances.)

When asking what sort of task each action instance should perform, FrameNet provides clues towards this answer. FrameNet labels actions with what are called **semantic frames**. Semantic frames can loosely be interpreted as scenes being carried out by agents and their props. The key idea is that any agent or prop within each semantic frame should be able to be swapped out with another object or prop belonging to the same frame, and the scene will be carried out in the same manner. For instance, lifting an apple and lifting an orange would belong to the same semantic frame. However, lifting an apple and lifting a boulder would not.

Further, FrameNet's concept of semantic frames are not limited in lexical scope to verbs. Because actions involve objects upon which the action is performed, FrameNet incorporates (among other information) nouns. Likewise, our system will also incorporate nouns. We will incorporate nouns into our model through "slots", or arguments that each action class and instance inherently possesses. These slots represent nouns related to verbs such as the verb's subject, direct object, and indirect object. When traversing the action hierarchy model, verbs will pass their corresponding nouns as arguments to action classes, which will pass their arguments to their children. The arguments passed can be swapped, added, or removed as necessary.

Our work splits from that developed for FrameNet in that we are looking to develop semantic frames that are robotics-focused. We are looking to create frames that will allow a robot to perform tasks such as retrieval, cleaning, and place setting. Our semantic frames will resemble those of FrameNet's in structure while being more focused on robotic tasks.

B. Implementation

In ROS, each action class is represented by an instance of a generic "action class" class in Python, which will be part of our developed framework. Classes will receive arguments from their parents (or from a verb) and will pass the arguments they receive to their children, possibly adding, removing, or swapping them. Action instances will also receive arguments in the same manner. Each action instance will have a corresponding actionlib server that will physically instantiate the instance. (Actionlib servers create a client-server model in which the client requests an action to be performed and the server performs it. Further, the client receives status reports and can interrupt or preempt the server's execution at any time.) Here, the actionlib client serves as our controller for performing actions, and the servers each perform an action specified by an action instance.

To search the graph created by the Action Hierarchy Model for answers, we are going to traverse the graph in a manner reminiscent of Prolog, i.e. with a depth-first search until we come across a possible solution which matches execution constraints. When a match is found, it will be executed. However, if that match fails to accomplish the given task, either by self-evaluation or by user preemption, the traversal

Fig. 2. Possible experiments include structure assembly, object retrieval, and place setting. (Center image: Popular Science. Right image: Josh Ellingson.)

will continue in an attempt to find another match. It may also be possible in the future to learn the best combinations and attempt them first.

IV. Evaluation

We will demonstrate the framework's capability of scaling to all possible ROS functionality by grounding a multitude of verbs using our action hierarchy model. To demonstrate this grounding, experiments will be performed in areas such as structure assembly, object retrieval, and place setting. We will use the PR2 platform to accomplish these tasks. Success will be measured by the scope of verbs and instructions grounded in our model and the variety of tasks successfully performed by our model.

V. Conclusion

Communication with robots through dialog allows users to interact with complex robots without diminishing either the necessary expressiveness of the commands or the abilities of the robot. The proposed action hierarchy model will facilitate this communication by enabling verbs to be grounded into concrete robot actions. Due to its domain agnosticism, this model can scale to ROS's capabilities. ROS offers community code support and provides a framework upon which to implement this model. We plan to develop our framework and evaluate it's abilities through mobile manipulation experiments using a PR2.

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

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

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

[5] 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.

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

[7] 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.

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