INTERACTIVE LEARNING OF VERB SEMANTICS TOWARDS HUMAN-ROBOT COMMUNICATION

By

Lanbo She

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2017

ABSTRACT

INTERACTIVE LEARNING OF VERB SEMANTICS TOWARDS HUMAN-ROBOT COMMUNICATION

By

Lanbo She

In recent years, a new generation of cognitive robots start to enter our lives. Robots such like ASIMO, PR2, and Baxter have been studied and applied in education and service applications. Different from traditional industry robots doing specific repetitive tasks in a well controlled environment, cognitive robots must be able to work with human partners in a dynamic environment which is filled with uncertainties and exceptions. It is unlikely to pre-program every type of knowledge (e.g., perceptual knowledge like different colors or shapes; action knowledge like how to complete a task) into the robot systems ahead of time. Just like how children learn from their parents, it's desirable for robots to continuously acquire knowledge and learn from human partners on how to handle novel and unknown situations. Driven by this motivation, the goal of this dissertation is to develop approaches that allow robots to acquire and refine knowledge, particularly, knowledge related to verbs and actions, through interaction/dialogue with its human partner. Towards this goal, this dissertation has made following contributions [i].

As a first step, we propose a goal state based verb semantics and develop a three-tier action/task knowledge representation. This representation on one hand supports the connection between symbolic representations of language and continuous sensori-motor representations of the robots; and on the other hand, supports the application of existing planning algorithms to address novel situations. Our empirical results have shown that, given this representation, the robot can immediately apply the newly learned action knowledge to perform actions under novel situations.

[i] This dissertation was supported by IIS-1208390 and IIS-1617682 from National Science Foundation.

Secondly, the goal state representation and the three-tier structure are integrated into a dialogue system on board of a SCHUNK robotic arm to learn new actions through human-robot dialogue in a simplified blocks world. For a novel complex action, the human can give an illustration through dialogue using robot's existing action knowledge. Comparing the environment changes before and after the action illustration, the robot can identify a goal state to represent the novel action, which can be immediately applied to new environments. Empirical studies have shown that action knowledge can be acquired by following human instructions. Furthermore, the results also demonstrate that step-by-step instructions lead to better learning performance compared to one-shot instructions.

To solve the insufficiency issue of applying the single goal state representation in more complex domains (e.g., kitchen and living room), the single goal state is extended to a hierarchical hypothesis space to capture different possible outcomes of a verb action. Our empirical results demonstrate that the representation of hypothesis space, combined with the learned hypothesis selection algorithm, outperforms approaches using single hypothesis representation.

Lastly, we address uncertainties in the environment for verb acquisition. Previous works rely on perfect environment sensing and human language understanding, which does not hold in real world situation. In addition, rich interactions between teachers and learners as observed in human teaching/learning have not been explored. To address these limitations, the last part presents a new interactive learning approach that allows robots to proactively engage in interaction with human partners by asking good questions to handle uncertainties of the environment. Reinforcement learning is applied for the robot to acquire an optimal policy for its question-asking behaviors by maximizing the long-term reward. Empirical results have shown that the interactive learning approach leads to more reliable models for grounded verb semantics, especially in the noisy environments.

ACKNOWLEDGMENTS

First and foremost I would like to thank and express my appreciation and gratitude to my advisor Dr. Joyce Y. Chai. I am greatly indebted to Dr. Chai for all I have learned from her in terms of conducting research, thinking critically, and how to be a good researcher. I also would like to thank Dr. Chai for her great insights about research in human-robot dialogue, which always light me up. Without her endless advice, inspiration, and guidance throughout my Ph.D. study, this work would have been impossible.

Also, I would like to thank my program committee members: Dr. Pang-Ning Tan, Dr. Xiaoming Liu, and Dr. Taosheng Liu. I appreciate their valuable feedbacks at every milestone of my Ph.D. process.

I would also like to express thanks to my collaborators at Michigan State University: Dr. Ning Xi and members from his lab, Dr. Yunyi Jia, and Yu Cheng, for their plentiful help with building my first physical dialogue agent, which initiated and established the foundation of my thesis work.

Thanks to members of the Language and Interaction Research Group: Dr. Changsong Liu, and Dr. Rui Fang for the valuable suggestions of conducting research when I was a new graduate student, Shaohua Yang, Qiaozi Gao, and Sari Saba-Sadiya for the discussion and enlightening comments, Kenneth Stewart for the help with building the Baxter system, and all my friends who made my time at MSU enjoyable.

Thanks to my parents for the selfless love and continued moral support throughout. Finally, the warmest thanks to my wonderful wife, Beibei Liu, who has always been supportive and encouraging. To my beloved family, I dedicate my dissertation work.

TABLE OF CONTENTS

LIST OF TABLES

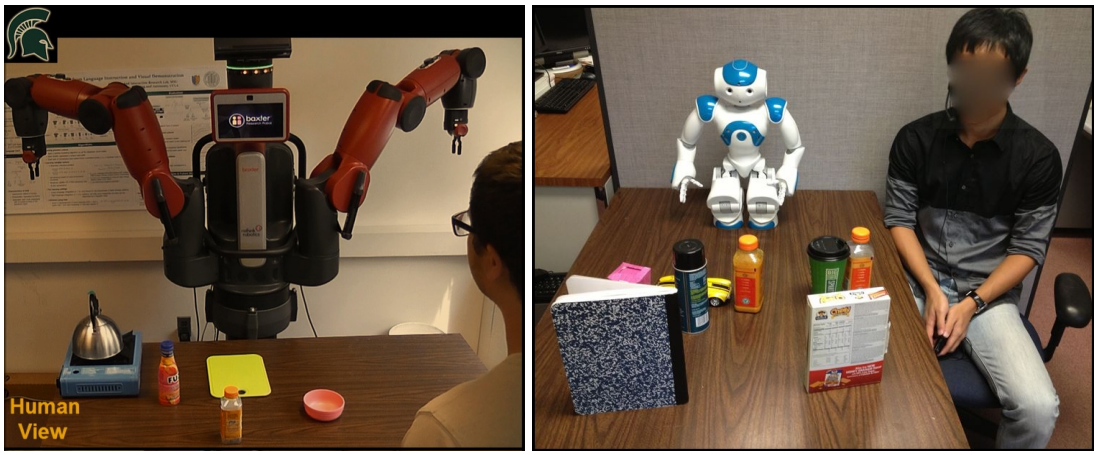
LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

In recent years, a new generation of cognitive robots have started to enter our lives. These intelligent agents can serve as assistants and companions, and work side-by-side with their human partners in joint tasks. Some existing products have been used in industry and different research institutes including but not limited to the Baxter, NAO, and ASIMO. Examples are shown in Figure 1.1. Because of their advanced perception and reasoning capabilities, these robots can be potentially utilized in a wide range of daily life applications: personal companion, house keeping (e.g., table cleaning, dish making), elderly care, and medical assistance in hospital.



(a) Baxter robot (b) NAO robot

Figure 1.1 Example intelligent robotic products applied in lab environments.

This new generation of cognitive agents is fundamentally different from the traditional robotic machines. Traditional robots (e.g., industrial robots) work in a well controlled environment. The

robot only needs to perform a specific task (e.g., each robotic arm on the assemble line always follows the same movement) and the tasks usually are deterministic. In these applications, the robots' knowledge can be pre-programmed and fixed in the system. However, a cognitive robot usually works in an uncontrolled environment. For example, suppose there is a robot working in a kitchen domain. The settings of each kitchen could be different. The objects in each kitchen could be different. And even for the same kitchen, the settings can be changed by time. It is undesirable if the robot needs to be re-programmed whenever it's deployed in a new environment. On the other hand, language is an important and natural method for human-human interaction. As a basic capability of human, being able to use language to establish communication and exchange information is desirable for a cognitive robot, such that the end users don't have to have expertise knowledge in robotics in order to work with a robot.

For a cognitive robot to work along with end users, it's impossible to program every situation into the system as the robot will always encounter "novel situations". The "novel situations" could include new objects the robot does not have any models of and new skills the robot have not been trained on.

Technologies to bridge the gap between natural language and the robot's perception (i.e., object recognition and referential grounding) have been widely developed. However, less work have been done to bridge the gap between robots and human language: specifically the action verbs used in our language. In this thesis, we are developing cognitive robotic systems that can continuously acquire the meaning of new verbs and learn how to perform these verbs through interacting with it human partner.

In the following sections of this chapter, we first identify the research challenges. Then, our contributions towards the goal of enabling a robot to learn new verbs are presented.

1.2 Challenges

As discussed in the Background section, it is important for the robots to automatically learn new verb/action knowledge from their human partners through interaction or dialogue. To address this issue, the following questions need to be answered:

1. Human language has a discrete and symbolic representation, but a robot has continuous representations for its movements. Where should the connections between the symbolic representation and the continuous representation take place (i.e., the higher-level concepts expressed by human language and lower-level primitive actions the robot is familiar with), so that human language can be used to direct the robot's movements?

2. When the robot learns new tasks from its human partner, how to represent the acquired knowledge effectively so that it can be applied in novel situations? For example, suppose a human directs a robot to "*fill the cup with milk*" by pouring the milk into the cup. And the robot is able to learn the semantics of the "*fill*" action after the demonstration. Is it possible that the acquired knowledge can be used if the working environment is changed? (e.g., if there's already water in the cup, the robot needs to empty the water in the cup before pouring milk.)

3. During human-robot dialogue, when the robot fails to perform the expected actions due to lack of knowledge, how should the human teach the robot? through step-by-step instructions or one-shot instructions? For example, when teaching the robot how to perform an action, the human can give one step at a time, wait for the robot to respond, then decides the next step instruction (e.g., either correct the robot if it makes mistakes, or continue with the next action). Or the human can give all the steps necessary to complete the verb all at once. What could be the advantages of each procedure?

4. Interaction is a collaborative process, where each participant attempts to coordinate his/her behavior with the other participant. Immediate feedback is an important factor helping us understand the others mental status during interaction or conversation. For a cognitive robot, how to model the dialogue such that the “collaboration" or the immediate feedback can be utilized to help with the knowledge acquisition process?

1.3 Contributions

Towards the goal of building cognitive agents that can learn verb knowledge from human-robot interactions, this thesis makes following contributions.

1. We developed a three-tier action knowledge representation that consists of high-level actions, primitive actions, and continuous trajectories.

    a) The upper-level captures the high-level actions acquired by learning from the human, which correspond to the concrete actions/verbs specified in human commands (e.g., the *grab* in “*Grab the blue block on the left.*", and the *stack* in “*Stack the red block on top of the green block.*", etc.). These high-level actions are represented as the desired goal states of the environment as a result of these actions. Instead of directly modeling a high-level action as a sequence of actions/steps specified by the human, capturing the desired goal states provides much flexibility to address novel situations as demonstrated in our empirical evaluations.

    b) The middle level defines primitive robotic operators such as *Open_Grip*, *Close_Grip* and *MoveTo* using AI planning operators [2] and directly links to the lower level. Motivated by the planning literature, the primitive robotic actions capture both pre-conditions

and effects of the robot's basic movements (e.g., move to a location, open/close gripper) which are directly linked to lower-level control functions.

c) The lower level connects to the physical arm and defines the trajectories of executing three atomic actions supported by the arm (i.e., *open_gripper*, *close_gripper*, *move*).

This three-tier representation allows the robot to automatically come up with a sequence of primitive actions by applying existing planning algorithms.

2. Based on the proposed three-tier verb representation, we implemented a dialogue system for action learning and further conducted an empirical study with human subjects. As a first step, we limited our investigation to a simple blocks world. In particular, we compared the dialogue based on the step-by-step instructions (i.e., one step at a time and wait for the robot's response at each step before going to the next step) with the one-shot instructions (i.e., give the instruction with all steps at once). Our empirical results have shown that the three-tier knowledge representation can capture the learned new action and apply it to novel situations. Although the step-by-step instructions resulted in a lengthier teaching process compared to the one-shot instructions, they led to better learning performance for the robot.

3. To address the over-fitting problem caused by representing verbs with a single hypothesis of goal state, we have developed an approach where each verb is explicitly represented by a hypothesis space of fluents (i.e., desired goal states) of the physical world. Given a human command, if there is no knowledge about the corresponding verb (i.e., no existing hypothesis space for that verb), the robot will initiate a learning process to ask human partners to teach how to accomplish this command. After the teaching, a hypothesis space of fluents for that verb frame will be automatically acquired. If there is an existing hypothesis space for the verb, the robot will select the best hypothesis that is most relevant to the current situation

and plan for the sequence of primitive actions. Based on the outcome of the actions (e.g., whether it has successfully executed the command), the corresponding hypothesis space will be updated. Through this fashion, a hypothesis space for each encountered verb frame is incrementally acquired and updated through continuous interactions with human partners. At this stage, to focus our effort on representations and learning algorithms, we applied our approach in a simulated environment where a virtual robot was used to interact with humans. Using an existing benchmark [3], our approach has demonstrated a significant performance improvement compared to approaches which only use simple hypotheses [3].

4. Despite the success of using goal state to represent verbs, the previous approaches work under strong assumptions of deterministic and perfect environment sensing and command understanding. However, this assumption does not hold in real-world applications. In addition, traditional approaches of model learning under noisy settings require a large set of parallel data (e.g., the language command and human demonstrations of execution). This is also infeasible, because it is undesirable to ask end users to repeat a demonstration multiple times only for the purpose of making the robot learn a single action. Motivated by the study of how humans learn new skills [4], we designed an agent that can actively ask questions to resolve its ambiguous understanding about the environments and language commands, such that reliable verb models can be acquired through a single human demonstration and language interaction. Specifically, if a command includes an action verb unknown to the agent, it will ask for a human demonstration, detect how will the environment is changed because of the human movement, ask questions and wait for human answers to resolve ambiguities, infer verb models, and apply the acquired models in the future interactions. The proposed approach uses reinforcement learning to allow the agent to acquire an optimal policy for its

question-asking behavior, such that the agent can achieve higher rate of successful learning by only asking a small number of questions. Our empirical results have shown that the proposed approach leads to more reliable models of grounded verb semantics, especially in the noisy environment.

1.4 Thesis Organization

The rest of chapters are organized as follows. Related research areas and works are introduced in Chapter 2. Then, in Chapter 3, core concepts used throughout this thesis are described in detail including verb goal state and symbolic planning, as well as a proposed three-tier action knowledge representation. In Chapter 4, an integrated dialogue system is described, which is built on the three-tier action representation and can acquire new verbs through human-robot dialogue. An empirical user study is conducted using this dialogue system. The experimental setups and evaluations are also described in Chapter 4. Chapter 5 introduces the concept of verb semantic hypothesis space, as well as an incremental learning procedure, to address the over-fitting problem caused by the previous representation. In Chapter 6, the hypothesis space is extended to utilize modeling the agent's question asking behavior to learn reliable grounded verb semantics under noisy environment understanding. Finally, we discuss future directions in Chapter 7.

Chapter 2

RELATED WORKS

Learning to perform actions through human-robot dialogue is related to multiple research areas, ranging from linguistic studies on verbs, action modeling, dialogue modeling, AI planning, to recent advances in grounding language to actions. In this chapter, we discuss some related works in these areas.

2.1 Linguistic Studies on Verbs

Verbs, especially those verbs used in language commands, usually signal concrete activities performed by an agent in an environment. In previous linguistic studies [5, 6], Hovav and Levin propose a theory to divide the action verbs into two types: ***manner verbs*** and ***result verbs***. Those manner verbs "specify as part of their meaning a manner of carrying out an action", while the result verbs "specify the coming about of a result state". Example verbs in those two categories are:

- Manner Verbs: *nibble, rub, laugh, run, swim ...*
- Result Verbs: *clean, cover, empty, fill, chop, cut, open, enter ...*

In our works, we are interested in the acquisition of ***result verbs***, which can be characterized by the changes of states. In [7], Kennedy develops a semantic typology of gradable predicates with respect to deverbal adjectives. Their analysis of predicates help us understand different dimensions of state changes, which is applicable to the result verbs. In [8], Schuler presents a library of verbs, paired with their logic or semantic forms. In this verb library, verbs are represented based on

argument frames (e.g., thematic roles which capture participants of an action). However, this frame based representation is insufficient to connect verbs with robot actions. As a result, even if it has a correct frame representation, the robot is still unable to perform the action. Motivated by these studies, our work focuses on developing verb semantic representation for result verbs that can “be executed" by a robot. Specifically, the goal state based verb semantics.

Over forty years ago, Terry Winograd developed SHRDLU [9] to demonstrate natural language understanding using a simulated block-moving arm. This system can answer questions, accept information, and execute commands through natural language dialogue. Winograd's system focuses on information exchanges of perceptual knowledge, like object properties or locations. One aspect Terry Winograd did not address, but mentioned in his thesis [9] as an important aspect, was learning new actions through natural language. Motivated by Winograd's early work, we started our initial investigation on action learning in a physical blocks world and with a physical robotic arm. The blocks world is the most popular domain used for planning in artificial intelligence. Thus it allows us to focus on developing mechanisms that, on one hand, connect symbolic representations of language with lower-level continuous sensorimotor representations of the robot; and on the other hand, support the use of the planning algorithms to address novel situations.

To enable human-robot communication and collaboration, recent years have seen an increasing amount of works which aim to learn semantics of language that are grounded to agents' perception [10, 11, 12, 13, 14, 15, 16, 17] and action [1, 3, 18, 19, 20, 21, 22, 23, 24, 25, 26]. Specifically for verb semantics, some previous works also tried to explore connecting action verbs with lower-level control systems. For example, Kress-Gazit et al. [27] utilized Linear Temporal Logic (LTL) to model a system's high-level task knowledge and support motion planning. A structured formal language is built to map command components to LTL fragments. However, different from natural language, the structured formal language is more of a text realization for robotic actions, which

requires specific training for the end users to use. In [28], Siskind developed a system to recognize events (i.e., spatial movement verbs) from image sequences, where verb semantics is specified by changes of force-dynamic relations between action participants. Another approach [19] learned a parser based on pairwise data of English commands and corresponding control language expressions. For this type of representations, they rely on collecting a large amount of training data ahead of time, which is not feasible for real world applications.

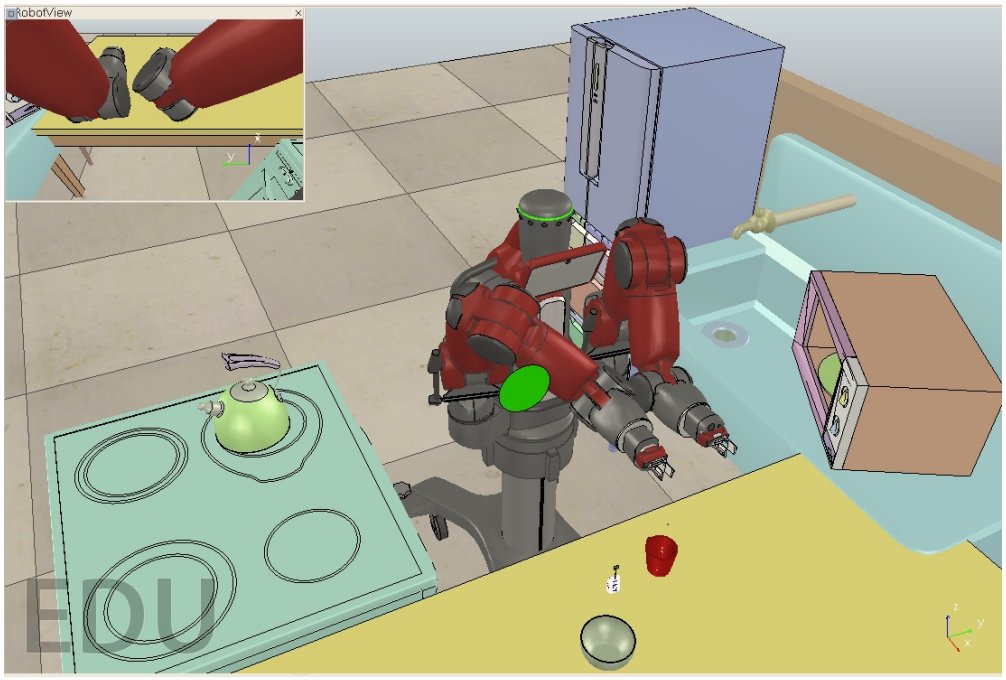


Figure 2.1 An example simulated kitchen environment. In this domain, a human can issue a command like *Microwave the cup*. The goal state for this command is $in(cup_3, microwave_1) \wedge state(microwave_1, is\text{-}on)$.

Another set of very relevant recent works tried to explore the connection between verbs and action planning [3, 18], which grounds higher-level commands such as “microwave the cup" to lower-level robot actions. In these works, each verb lexicon is represented as desired goal states of the physical world as a result of the corresponding actions. For instance, the goal of command “*Microwave the cup.*" is represented as $in(cup_3, microwave_1) \wedge state(microwave_1, is\text{-}on)$. Such representations are learned based on example actions demonstrated by humans. Once acquired,

these representations will allow agents to interpret verbs/commands issued by humans in new situations and apply action planning to execute actions. Their empirical evaluations once again have shown the advantage of representing verbs as change of state for the robotic systems. Given its clear advantage in connecting verbs with actions, our work also applies the state-based representation for verb semantics. Different from these previous works, we represent verb semantics through a hypothesis space of goal states. In addition, we present an incremental learning approach for inducing the hypothesis space and choosing the best hypothesis. Also, we have developed new approaches which go beyond learning from demonstrated examples but explore rich interaction between humans and agents to acquire models for grounded verb semantics.

2.2 AI Action Modeling

In AI literature on action modeling, action schemas are defined with preconditions and effects. Thus, representing verb semantics for action verbs using resulting states can be connected to the agent's underlying planning modules. Some works in this area learn action models from example plans. For example, in [29, 30], the authors propose approaches to learn planning operators by observing action sequence solutions and further refine the operators through practice. And in [31], given a set of plan examples, the operator models are learned by translating the problem of finding the optimal set of operators to a weighted MAX-SAT problem. In their settings, each operator is a deterministic and unique element of the environment dynamic. However, in applications of interacting with human, the representation for a verb can be a set of hypotheses.

2.3 Learning to Follow Instructions

Learning to follow instructions has received an increasing attention in recent years. Previous works have applied supervised learning or reinforcement learning based on large corpus of data. Specifically, following efforts have been made toward this direction.

Branavan et al. [32, 33, 34] proposed a set of approaches utilizing reinforcement learning to map language instructions to sequences of primitive executable operations. By assuming access to a reward function, the system can acquire an optimal policy through continuously interacting with the environment. In [35, 36, 37], Tellex and Kollar et al. proposed models for the robot to interpret natural language commands towards navigation and manipulation tasks. Specifically, a language command is translated to a probabilistic graphical model which infers the likelihood of a sequence of primitive robot actions. The graphical models are trained with a set of parallel data. In another approach, Chen [38] et al. proposed a framework to interpret and generate language descriptions for events using perceptual context as supervision. A set of paired data is used to train the model, where each data includes a stream of descriptive textual comments and a sequence of events. The framework is demonstrated in a system that learns to sportscast robot soccer games in English and Korean.

However, the major problem here is that when the robot is deployed in the real world to work with people, it is not feasible to provide a large amount of training data relevant for the current situation. It is also not ideal to engage the robot in a lengthy exploration process given time constraints. These limitations makes the previous approaches inadequate in time-critical situations. What's available is the human partner co-present with the robot. Thus it is desirable for the robot to directly learn from its human partner. And it is desirable that humans can engage in a natural language dialogue to teach robots new skills.

2.4 Learning from Demonstration

In the robotics community, Learning from Demonstration (LfD) or Programming by Demonstration (PbD) is another related area towards the goal of enabling robots to perform new actions/tasks. The main idea of LfD is that users can teach robots new tasks without programming. For example, a human can manipulate the robot to demonstrate a task (e.g., using a remote controller, or moving the robot to perform), or a human can perform the task and let the robot watch the demonstrations. In Akgun's work [39], a human participant moves the arm of a Simon robot to give multiple demonstrations of a task (e.g., pouring and placing). In each demonstration, a sequence of key-frame are extracted and the arm configurations in those key-frames are recorded. The key-frame arm configurations from multiple demonstrations together form a set of clusters in the configuration space. The centers of these clusters are the key poses of the task. And, by sequentially following those key poses, the robot should be able to perform the task. A comprehensive survey of robot Learning from Demonstration can be found in [40]. Different from these LfD works, here we also investigate the use of natural language dialogue for task learning.

2.5 Learning from Dialogue

Another closely related area is learning through dialogue. Different from making a system that learns by itself through corpus data (e.g., pair labeled data or unlabeled data), dialogue or interactions provide better opportunities for an intelligent agent to get more help from the human partner, in a more direct and immediate fashion. It worths investigating what could be the better dialogue/interaction model to fully utilize these additional supervisions in the agent's learning or inference process. Specifically, *Common Ground* is a concept proposed in dialogue study [41] [42], essential to the success of communication. In conversation, participants coordinate their mental

states based on their mutual understanding of their intentions, goals, and current tasks [43]. Clark and colleagues have further characterized this process by defining *contributions*, the basic elements of a conversation [44], such that the grounding process can be viewed as a collaborative process where the speaker and the addressee make extra efforts to collaborate with each other to reach a mutual understanding [45]. The notion of common ground and communication grounding have been investigated extensively from human-human communication [42, 46], to computational models for spoken dialogue systems [47], and more recently to human-robot interaction [48, 49, 50, 51, 52], symbol grounding [53] and feedback and adaptation [54] in human-robot dialogue.

Most related to our work here, some recent works have also explored the use of natural language and dialogue to teach agents actions. For example, [55] applied natural language dialogue technology to teach an artificial agent (web-based agent) actions (e.g., order books, find restaurant, etc.). However this previous work only needs to ground language to interface widgets (i.e., symbolic representation) without considering lower-level continuous actions as in robots. Related to human-robot interaction, natural language has been applied for the robot to learn new actions [56][57][58], parsing strategies[19], learning symbols[59]. Different from these earlier works, our work focuses on action learning that connects high-level language with lower-level robotic actions using goal state based representations and symbolic planning.

Our work was also motivated by previous cognitive studies [60] on how people learn as well as recent findings on robot skill learning [4]. One of the principles for human learning is that "learning is enhanced through socially supported interactions". Studies have shown that social interaction with teachers and peers (e.g., substantive conversation) can enhance student learning experience and improve learning outcomes. In recent work on interactive robot learning of new skills [4, 61], researchers identified three types of questions that can be used by a human/robot student to enhance learning outcomes: 1) *demonstration query* (i.e., asking for a full or partial demonstration of the

task), 2) *label query* (i.e., asking whether an execution is correct), and 3) *feature query* (i.e., asking for a specific feature or aspect of the task). Inspired by these previous findings, in the last part of our work, we explored interactive learning to acquire grounded verb semantics. In particular, we aimed to address when to ask what questions during interaction to improve learning.

Chapter 3

VERB GOAL STATE AND SYMBOLIC PLANNING

3.1 Introduction

A new generation of robots have emerged in recent years which serve as humans' assistants and companions [62]. However, due to significantly mismatched capabilities (e.g., perception and actions) of humans and robots, natural language based communication between them will be difficult. First, the robot's representation of its perceptions and actions are continuous and numerical in nature. But human language is discrete and symbolic. For the robot to truly understand human language and thus take corresponding actions, it needs to first ground the meanings of human language to its own sensorimotor representation of its perception and action[27]. Second, the robot may not have complete knowledge about the shared environment and the joint task. It may not be able to connect human language to its own representations given its limited knowledge. Thus it is important for the robot to continuously acquire new knowledge through interaction with humans and the environment.

To address these challenges, we have developed techniques to bridge the gaps of perception and actions to support situated human-robot dialogue [63]. While most of our previous work have focused on the perceptual differences [64][65], in this chapter we investigate the gap of action. In particular, we are developing techniques that allow humans to teach new high-level actions to the robot through natural language instructions. This chapter focuses on details of the representation

A significant portion of this chapter was published in the following paper: Lanbo She, Yu Cheng, Joyce Y. Chai, Yunyi Jia, Shaohua Yang, Ning Xi. Teaching Robots New Actions through Natural Language Instructions. In the 23rd IEEE International Symposium on Robot and Human Interactive Communication IEEE RO-MAN'14, Edinburgh, UK, 2014, pages 868–873.

to connect symbolic language with continuous robotic action.

One important issue in action learning through natural language instructions is the representation of the acquired action in the robot's knowledge base so that it can be effectively applied in novel situations. To address this issue, we develop a framework for action representation that consists of primitive actions and high-level actions. Motivated by the planning literature [2], the primitive robotic actions capture both pre-conditions and effects of the robot's basic movements (e.g., move to a location, open/close gripper) which are directly linked to lower-level control functions. Our high-level actions [i] correspond to the concrete actions/verbs specified in human commands (e.g., the *grab* in "*Grab the blue block on the left.*", and the *stack* in "*Stack the red block on top of the green block.*", etc.). These verbs tend to denote concrete activities performed by an agent. In our work, they are modeled by the desired goal states of the environment as a result of these actions. The goal states can be automatically acquired after following step-by-step instructions by the human. Instead of directly modeling a high-level action as a sequence of actions/steps specified by the human, capturing the desired goal states provides much flexibility to address novel situations as demonstrated in our empirical evaluations. Given a specific robot, its basic capabilities (e.g., basic movements) are fixed. But the verbs a human would use in the commands are potentially unlimited which cannot be pre-programmed and have to be continuously acquired. For works presented in this chapter, we assume the robots' primitive capability (i.e., the primitive planning operators and low-level continuous trajectories) are known and fixed. And when we say acquire new verbs, we always mean learning the semantics of verbs specified in the human language commands.

In the following sections, we first introduce the main concepts: 1. action goal state representation; 2. symbolic planning, used in the representations is presented. Then, the proposed three-tier action knowledge representation is explained in detail. And finally the conclusions are given.

[i] In the following chapters, for high-level actions, the terms complex "action" and "verb" are used interchangeably.

3.2 Representing Action with Goal State

Concrete verbs usually represent concrete activities performed by an agent, either a human or a robot. In linguistic theory about lexicon semantics of concrete actions, these verbs could be categorized as two categories: RESULT VERBS and MANNER VERBS [5, 6]. The MANNER VERBS mainly "*specify as part of their meaning a manner of carrying out an action*" [6] which concern the form or manner of the activities denoted by the verbs. While the RESULT VERBS typically "*specify the coming about of a result state*" [6] which denote the change of states of the objects manipulated during the activities. Example verbs of these two categories are [5, 6]:

a) MANNER VERBS: nibble, rub, scribble, sweep, flutter, laugh, run, swim,...

b) RESULT VERBS: clean, cover, empty, fill, freeze, kill, melt, open, arrive,...

In our work, we study the RESULT VERBS in particular, which concern more about the states resulted by executing the verb. Specifically, the goal state of a verb is represented by a conjunction of state units/terms which will be true (i.e., entailed/satisfied by the final environment) if the verb is executed. Each of these terms represents either certain state aspect of an object or a specific relation (e.g., spatial relation) between objects. Examples are like:

Comparing with using procedures or action sequences to represent high-level verbs, using goal state to represent language verbs has following advantages.

First, using goal state to represent verb semantics achieves higher generality. In particular, this means the goal state of a verb acquired through one learning instance can be applied to multiple novel situations. As an example, suppose we want to teach a robot how to *pick up* an object x with a set of primitive actions (i.e., *moveto* and *open/close gripper*). During learning, the example *pick up* can be achieved through a sequence of primitive actions $moveto(x), close\_gripper, moveto(air)$,

Verb	Goal State Representation	Interpretation
grab(x)	$G\_Close \wedge In\_G(x)$	To grab an object x means to reach a state where the gripper is close, and object x is in the gripper.
stack(x, y)	$G\_Open \wedge On(x, y)$	To stack object x on y means, after the action, the gripper should be open and object x should on top of another object y.
fill(x, y)	$Grasping(x)$ $\wedge Has(x, y)$ $\wedge \neg On(x, o_1)$	Filling object x with y means the robot should grasp object x, x should have y inside, and x is not on top of any other object o_1.
boil(x)	$Status(x, WTempHigh)$ $\wedge On(x, Stove)$ $\wedge Near(Robot, Stove)$	To boil an object x means to achieve a state where the object x will have high temperature water inside, the object should be on a stove, and the robot is near a stove.

Table 3.1 Examples of using goal states to represent action verbs.

whereas in testing the object x could be blocked by another object where the *pick up* cannot be achieved by directly executing this specific teaching action sequence. However, when using goal state to represent a verb, the acquired semantic for *pick up* from the same learning instance is: the gripper is closed and the object x is in the gripper. This type of semantic representation for *pick up* is applicable in different situations even when object x is blocked. And the environment dependent action sequence can be calculated through symbolic planning with knowledge of the robot's primitive actions.

Second, using goal state to represent verb semantics is more compact, and at the same time more close to the definition of RESULT VERBS in linguistic literature. Take the above *pick up* action as an example, when using action sequence to represent the verb, the lengths of the action sequences vary from 3 to more than 10 (e.g., when the target object is crowded by a set of obstacles). While, as shown above, using goal state to represent *pick up* takes only two state terms: 1. the gripper is closed and 2. the object x is in the gripper.

3.3 Symbolic Planning

The Classical Planning or Symbolic Planning–*devising a plan of action to achieve one's goals–is a critical part of AI* [66]. In essence a symbolic planner is a problem-solving agent, whose goal is to find a solution (i.e., a sequence of primitive operators/actions) to change the state such that certain constraints are satisfied. In general, *domain* and *problem* are two main concepts for symbolic planning, where the *domain* defines the representations and dynamics of a self-contained world the problems are embedded in and a *problem* essentially describes constraints that need to be satisfied. Currently, the *STRIPS* [2] and *Planning Domain Definition Language (PDDL)* [67] are commonly used as a standard language to describe domains and problems.

3.3.1 Defining Symbolic Planning

In symbolic planning, a state is represented by a conjunction of grounded fluents, where each fluent describes certain aspect of the status of the grounded object. A planning *domain* is defined as a set of action schemas, where each schema is a primitive operator the planner can use to generate action sequence. We use an example from Russell's well known AI textbook [66] to illustrate basic concepts in classic planning. The example blocks world domain, including 2 primitive schemas: 1. $Move(b, x, y)$; 2. $MoveToTable(b, x)$, is shown in Figure 3.1.

$Move(b, x, y)$
Precondition: $On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge Block(y) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$
Effect: $On(b, y) \wedge Clear(x) \wedge \neg Clear(y) \wedge \neg On(b, x)$

$MoveToTable(b, x)$
Precondition: $On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x)$
Effect: $On(b, Table) \wedge Clear(x) \wedge \neg On(b, x)$

Figure 3.1 An example of simplified blocks world.

Each action schema is composed by three sections. The first part is the action name and a set of arguments (e.g., the first action schema in Figure 3.1 is named $Move$ and it has three arguments b, x, and y). The second part is a precondition (i.e., Precondition in the figure) represented by a conjunction of state terms, which means this schema is applicable in a state if and only if all the terms specified in the precondition can be entailed in that state. The third part is a set of effects representing the outcomes of this action (i.e., the changes this action will make to the state). Usually, operator effects consist of two subcategories: positive effects (i.e., effects that will be added to the original state after applying the action) and negative effects (i.e., effects that will be removed from the original state). For example, the $Move(b, x, y)$ action has four effects: two of them are positive effects (i.e., $On(b, y)$ meaning, after moving b from x to y, b is on top of y. $Clear(x)$ meaning there will be nothing on top of x) and two negative effects (i.e., $\neg On(b, x)$ meaning b is no longer on x. $\neg Clear(y)$ meaning y has something on top of it). The set of schemas describes the dynamics of a domain.

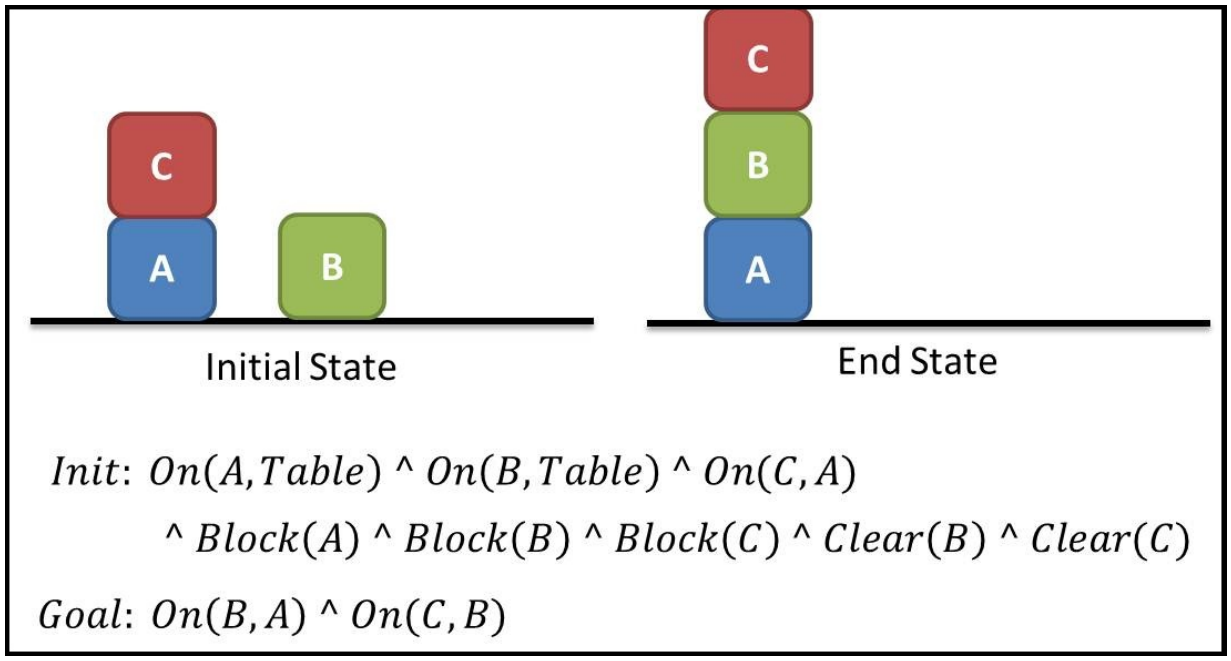


Figure 3.2 An example problem in the simplified blocks world.

A specific planning *problem* to in a *domain* is defined with an *initial state* (i.e., the beginning state of the problem) and a *goal state*. Similar to the preconditions and effects in planning operators, both of the initial state and goal state are represented by a conjunction of state terms/literals. And a planning problem is treated as solved if the planner finds a sequence of grounded primitive operators that ends in a state that the goal state can be entailed. An example problem from in the blocks world domain is shown in Figure 3.2, where the goal is to move B onto A, and move C onto B. And finally, this problem can be solved by the following action sequence: $MoveToTable(C, A), Move(B, Table, A), Move(C, Table, B)$.

3.3.2 Algorithms for Symbolic Planning

Given a planning domain and a specific problem, different algorithms have been studied to find a plan (i.e., a sequence of grounded primitive actions) to solve the problem. Traditional planning algorithms including search based, planning graph base, and translating to a SAT problem [66]. And since 1998 the first International Planning Competition, more efficient algorithms are proposed to solve large scale planning problems [68, 69, 70]. In this chapter, a simplified blocks world is used as the testing domain, and a forward space-searching based problem solver[ii] is utilized to solve problems.

3.4 A Three-tier Action Knowledge Representation

The three-tier action knowledge representation–`Action Knowledge Base`– is illustrated in Figure 3.3. At the top level is the *High-level action knowledge*. This top level stores system knowledge about verbs that could be used in human language commands. As discussed in the previous Section 3.2, these verbs are represented with a corresponding resulting goal state. Some of verbs in this

[ii]The planner implementation can be found in https://bitbucket.org/malte/pyperplan

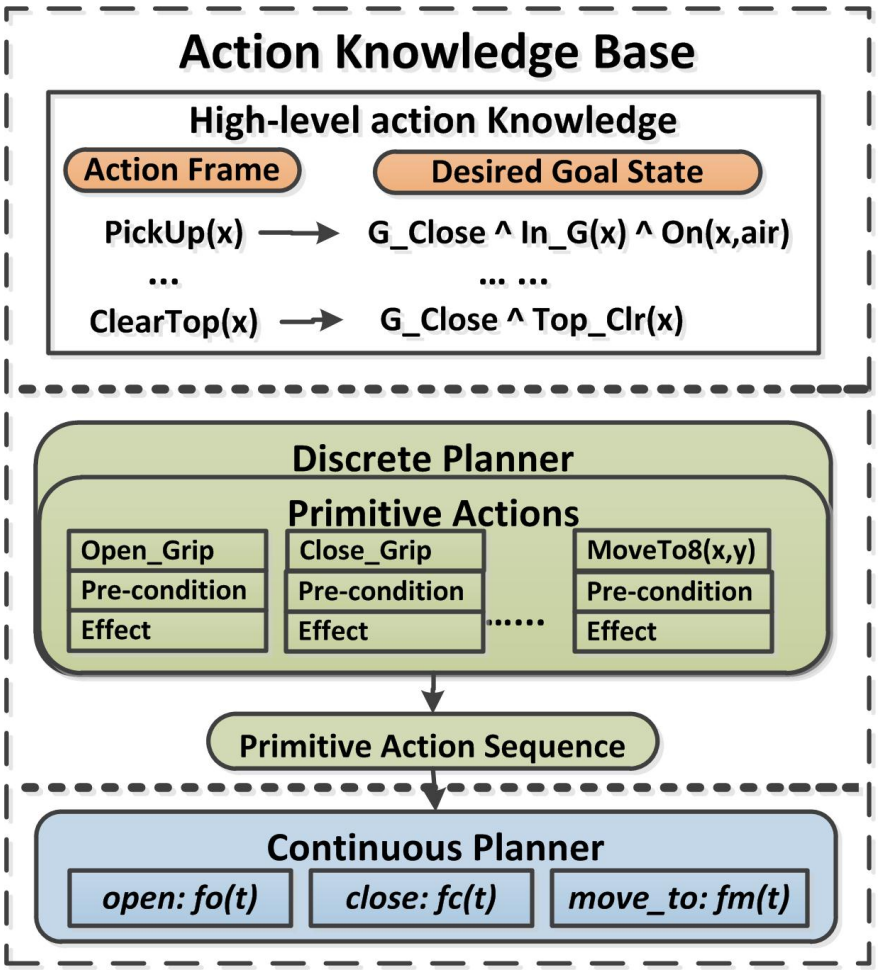


Figure 3.3 The three-tier action knowledge representation.

level are initialized by the designer, while most of the verbs should be acquired during real-time interaction (e.g., when interacting with users, there can always be verbs that the users may use but are novel/unknown to the robot) which are used for further interactions. In the middle level is a *Discrete Planner*, like *STRIPS* or *PDDL* planner, maintaining domain dynamics. As discussed in Section 3.3, the planning domain is characterized by a set of action schemas, where each is a primitive system action represented by preconditions and effects. These primitive actions are the robot's basic capabilities in the domain, which are pre-defined and fixed. The bottom level is a *Continuous Planner* which is responsible for calculating physical trajectories to realize each of the primitive actions performed by the robot.

3.4.1 High-level Action Knowledge

This level captures system knowledge of high-level actions specified in the human language that are known to the robot. Examples of such action are like $PickUp(x)$ and $ClearTop(x)$ shown in Figure 3.3. In the current simplified blocks and robotic arm domain, each of these actions is modeled by a desired goal state:

$$\mathcal{A}_i(c_1...c_{k_i}) = p_1(c_1...c_{k_i}) \wedge ... \wedge p_{n_i}(c_1...c_{k_i}) \wedge p_{grip}$$

$\mathcal{A}_i \in A$ is the name of a high-level action known to the robot. Each c_i is an object (e.g., block) in the robot's view. $p_i \in \mathcal{P}$ is one of the predicates defined in the domain. And the $p_{grip} \in \{G\_Open, G\_Close\}$ specifies the status of the robotic gripper (i.e., whether the gripper is open or not). An example complete representation of "$Pickup(x)$" is shown in Figure 3.3 which is described as "$G\_Close \wedge In\_G(x) \wedge On(x, air)$" meaning after picking up x the gripper will be closed, object x will be in the gripper, and x will be in the air.

3.4.2 Discrete Planner

The middle level action knowledge is used to automatically generate a plan (i.e., a sequence of planning operators) to achieve the goals specified by the high-level actions. Specifically, the *Discrete Planner* here is implemented as a *STRIPS* planner[ii][2], which is a well-known AI planner. In *STRIPS*, a domain can be formally defined as a 2-tuple $\mathcal{D} =< \mathcal{P}, \mathcal{O} >$, where $\mathcal{P}$ is a finite set of predicates to symbolically represent the world and $\mathcal{O}$ is a set of legal operators distinguishing changes of the domain world. A planning problem is defined as another 2-tuple $\mathcal{Q} =< \mathcal{I}, \mathcal{G} >$, where $\mathcal{I}$ represents the world status at the beginning of the problem (i.e., the initial state) and $\mathcal{G}$ is the desired goal state that should be achieved (i.e., the goal state).

In a planning domain, each state is represented by a conjunction of fluents including the status and relations of any objects in the scene. Each fluent here corresponds to a grounded term $p(c_1, c_2, ..., c_k)$, where $p \in \mathcal{P}$ and c_i is an object in the domain. In the simplified blocks world domain, the $\mathcal{P}$ set captures two dimensions of object status:

- Robotic arm states:

    1. *G_Open/Close* stands for whether the gripper is open or closed.
    2. *G_Full/Empty* represents whether the gripper has an object in it.
    3. *G_At(x)* describes where the arm locates.

- Object states:

    1. *Top_Uclr/Clr(x)* means whether the block *x* has another block on its top.
    2. *In/Out_G(x)* stands for whether it's within the gripper fingers or not.
    3. *On(x,y)* means object *x* locates on *y*.

Each operator $o \in \mathcal{O}$ is defined as a set of pre-conditions and effects. Both pre-conditions and effects can be any conjunctions of fluents or the negated fluents. An instance of operator o is only applicable when the pre-conditions can be entailed in the world state. And after accomplishing the operation, the world will change by adding the positive fluents and deleting the negated fluents in effects. In our blocks world domain, the operators include *Open_Grip*, *Close_Grip* and 8 types of *MoveTo*. The 8 *MoveTo*s are used to distinguish different situations of arm position changes, which can be categorized as two sets:

- When the gripper is open:

1. (m_{oe}_obj): The gripper has no object between fingers (empty) and move destination is an object.

2. (m_{oe}_loc): The gripper is empty and moves to locations other than objects (e.g., table and air).

3. (m_{of}_obj): The gripper is open but there's an object between the fingers (full) which can only occur when the object is supported, the destination is an object.

4. (m_{of}_loc): The gripper is full and destination is locations other than objects.

- When the gripper is closed:

1. (m_{cf}_obj_1_obj_2): The gripper is closed and full, it moves the in-hand object o from the top of object obj_1 to the top of another object obj_2.

2. (m_{cf}_obj_1_loc): Similar to the m_{cf}_obj_1_obj_2 but the destination is a location.

3. (m_{cf}_loc_1_loc_2): The gripper moves the in-hand object from one location loc_1 to another location loc_2.

4. (m_{cf}_loc_obj_2): The gripper moves the object from one location to the top of an object obj_2.

3.4.3 Continuous Planner

The lowest level of our knowledge representation is a *Continuous Planner*. It is directly connected with the robotic control system and characterizes the basic operations the robot can perform. In general, this low level representation can include any atomic robotic operations, considering different types of physical robots. For the SCHUNK arm we used, the basic capability can be categorized as three atomic operations: *open* (i.e., open gripper), *close* (i.e., close gripper) and *move_to* (i.e.,

move to certain location). Each of these three actions corresponds to a pre-defined end-effector trajectory parameterized by time *t*. Specifically, the *open/close* is a function *fo(t)/fc(t)*, which only controls the distance between two gripper fingers attached to the arm. For the *move_to*, given a destination coordinate, function *fm(t)* calculates the trajectory for the end-effector to reach the location but does not change the distance of the gripper fingers. These functions are further translated to control commands through inverse-kinematics and sent to each motor to track the trajectory.

3.5 Conclusion

In this chapter, we focus on how to represent high-level actions so that the learned actions can be applied in different situations. Specifically, we have developed a three-tier action knowledge representation for the robotic arm. The lower level connects to the physical arm and defines the trajectories of executing three atomic actions supported by the arm (i.e., *open_gripper*, *close_gripper*, *move*). The middle level defines primitive operators such as *Open_Grip*, *Close_Grip* and *MoveTo* in the fashion of the traditional AI planner [2] and directly links to the lower level. The upper-level captures the high-level actions acquired by learning from the human. These high-level actions are represented as the desired goal states of the environment as a result of these actions. This three-tier representation allows the robot to automatically come up with a sequence of lower-level actions by applying existing planning algorithms.

Chapter 4

VERB GOAL STATE LEARNING THROUGH HUMAN-ROBOT DIALOGUE

4.1 Introduction

Through discussions of a three-tier action knowledge structure in Chapter 3, we have seen that representing high-level commands with the corresponding goal states could enable the robot to follow natural language commands given by humans. Nevertheless, the process of acquiring these goal state representations from interactions with humans is rather complex and many different kinds of exceptions could occur. For example, when learning a new action, the robot could encounter another unknown action. Thus, it is important to support dialogue and keep track of dialogue discourse to understand relations between different actions. Furthermore, besides actions, the robot may not perceive the environment perfectly. The human and the robot will need to maintain a common ground in order for teaching/learning to be successful. To address these issues, from this chapter we will discuss how to extend the goal state representation with collaborative dialogue to support automated learning of new actions.

To address this issue, this chapter firsty describes our recent effort on action learning through dialogue [21]. As a first step, we limit our investigation to a simple blocks world motivated by Terry Winograd's early work [9]. By using an industrial robotic arm (SCHUNK) in this small world, we are interested in addressing the following questions. When the robot fails to perform the expected actions due to the lack of knowledge, what should be the procedure to acquire this

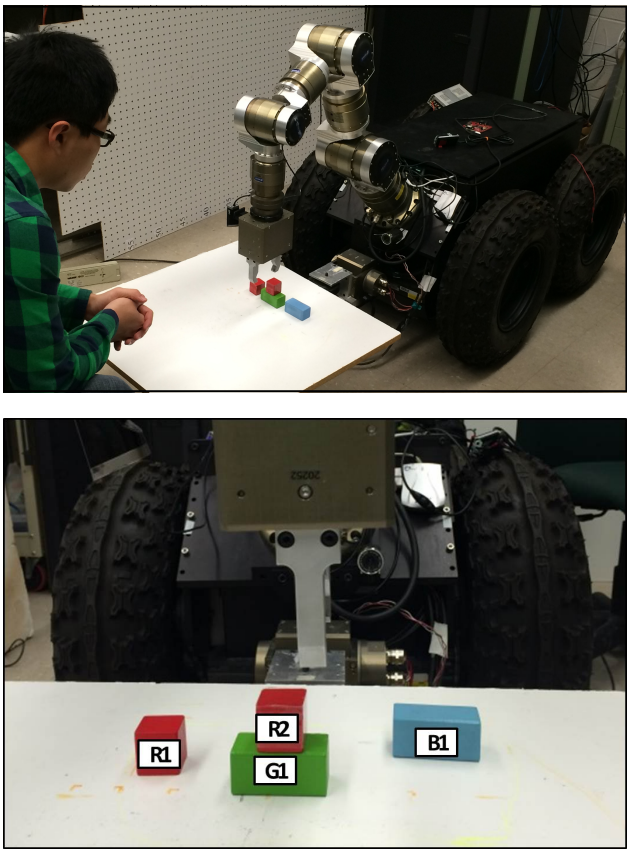
The majority of this chapter was published in the following paper: Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y. Chai, Ning Xi. Back to the blocks world: Learning new Actions through situated human-robot dialogue. In Proceedings of the SIGDIAL 2014 Conference, the 15th Annual Meeting of the Special Group on Discourse and Dialogue, Philadelphia, PA, USA, 2014, pages 89–97.

new action? Secondly, during human-robot dialogue, how should the human teach the robot new actions? through step-by-step instructions or one-shot instructions?

To answer these two questions, we implemented a dialogue system for action learning based on the three-tier action representation. And an empirical study with human subjects is further conducted. Different from previous work on learning through demonstrations [71], here we rely on natural language instructions to teach robots new actions. In particular, we compared the dialogue based on the step-by-step instructions (i.e., one step at a time and wait for the robot's response at each step before going to the next step) with the one-shot instructions (i.e., give the instruction with all steps at once). Our empirical results have shown that the three-tier knowledge representation can capture the learned new action and apply it to novel situations. Although the step-by-step instructions resulted in a lengthier teaching process compared to the one-shot instructions, they led to better learning performance for the robot.

4.2 Dialogue System

We developed a dialogue system to support learning new actions. An example learning setup and a typical teaching dialogue are shown in Figure 4.1, in which a SCHUNK arm is used to manipulate blocks placed on a surface. In H_1, the human starts to ask the robot to stack the blue block (i.e., B_1) on top of the red block (i.e., R_1). The robot does not understand the action "stack", so it asks the human for instructions. Then the human provides detailed steps to accomplish this action (i.e., H_2 to H_8) and also observes the robot's response in each step. Note that during this process, another unknown action (i.e., "grab" as in H_2) is encountered. The robot thus needs to learn this action first. The robot is able to keep track of the dialogue structure so that actions and sub-actions can be learned accordingly. Once the robot receives a confirmation from the human that the corresponding action is successfully performed (i.e., H_6 and H_9), it acquires the new action and explicitly rep-



H1: *Stack the blue block on the red block on your right.* R1: *How to stack, could you teach me?*
H2: *Grab the blue block.* R2: *What do you mean by grab?*
H3: *Open your gripper.* R3: *Ok.*
H4: *Move to the blue block.* R4: *Sure.*
H5: *Close gripper.* R5: *Alright then.*
H6: *Now you achieve the grab action.* R6: *Ok, got it.*
H7: *Move the blue block to the top of the right red block.* R7: *Understood.*
H8: *Open gripper.* R8: *Sure.*
H9: *Ok, now you are done with the stack action.* R9: *Ok.*

Figure 4.1 An example setup and dialogue. Objects are marked with labels only for the illustration purpose.

resents it in its knowledge base for future use. Instead of representing the acquired knowledge as specific action steps as illustrated by the human, the acquired verb is represented by the expected final state, which represents the changes of environment as a result of the action. The acquired new action can be directly applied to novel situations by applying planning algorithms. Figure 4.2 shows the system structure. Next we explain main system modules in detail.

4.2.1 Natural Language Processing

Human language is processed by a set of NLP modules to create a semantic representation that captures the meaning of each human utterance (e.g., intent of the utterance, focus of attention, etc). An example semantic representation of "H1: *Stack the blue block on the red block on your right.*" is

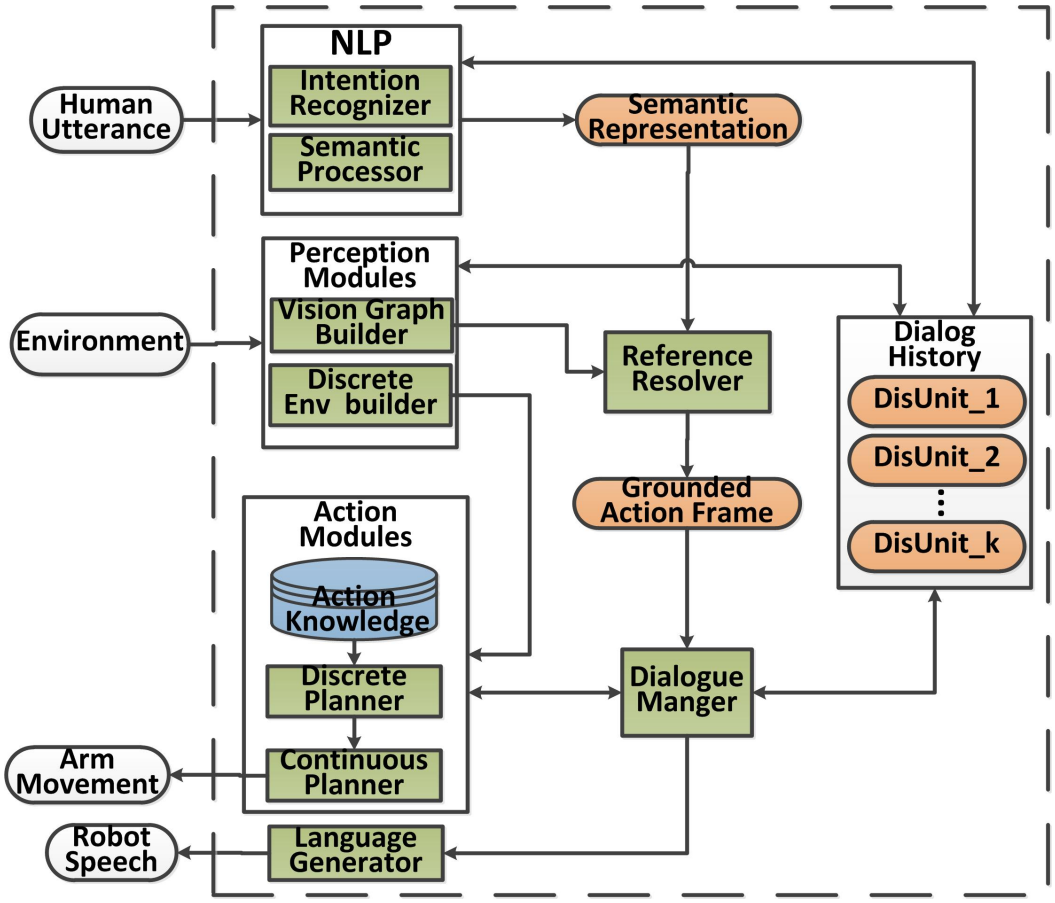


Figure 4.2 System Architecture

shown on the left side of Figure 4.3. In particular, two types of language informations are captured: 1. `Intention information`; and 2. `Action related information`.

4.2.1.1 Intent Recognizer

An `Intent Recognizer` is applied to identify the human intent (represented by `Intent`) such as whether it is a command requesting the robot to perform certain actions (i.e., `Command`) or a confirmation of the teaching illustration completion (i.e., `Confirm`). Currently, we only implemented a simple rule-based approach for intent recognition. Since our domain is quite narrow (mainly involving issuing commands and making confirmation), it appears sufficient.

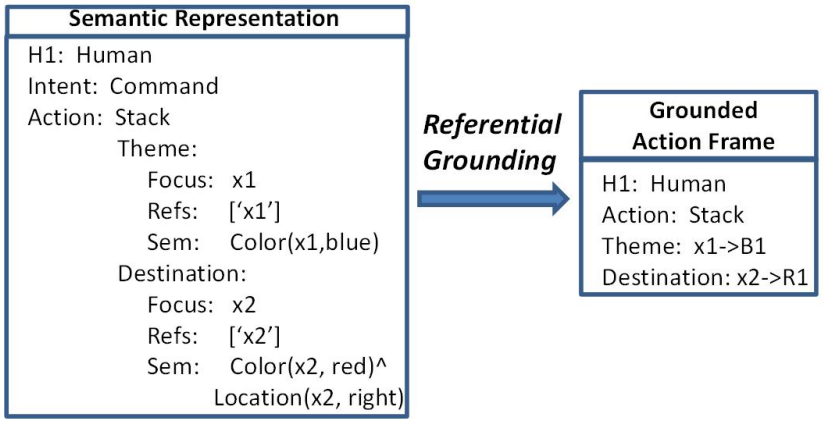


Figure 4.3 Example semantic representation and action frame for the human utterance "*stack the blue block on the red block on your right.*"

4.2.1.2 Semantic Processor

Besides human intent, a `Semantic Processor` is applied to identify the main verbs (represented by `Action`), as well as linguistic entities (i.e., `Theme` and `Destination`) and their properties mentioned in the utterance (e.g., color, location, or spatial relations). Specifically, a parser based on combinatory categorial grammar (CCG) [72] is applied [i]. We have defined a set of basic CCG lexicon rules, which covers key vocabulary in our domain, such as object colors, locations, spatial relations and so forth. Given a human utterance, our CCG parser repeatedly searches for the longest sequence covered by the grammar until the end of the utterance. As a result, the `Semantic Processor` creates a list of linguistic entities introduced by the utterance, and a list of first order predicates specifying the properties and relations between these entities. For the sentence shown in Figure 4.3, its main `Action` is *Stack*. A `Theme` object is recognized (i.e., named by x_1) which has a property of color *blue*. A `Destination` is also recognized (i.e., named x_2) which is a *red* object located on the *right*.

[i] We utilized *OpenCCG*, which could be found at: http://openccg.sourceforge.net/

4.2.2 Perception Modules

Besides interpreting human language, the robot also continuously perceives the shared environment with its camera. Objects in video frames are recognized through a real-time object recognition system MOPED [73] (`http://personalrobotics.ri.cmu.edu/projects/moped.php`) available from ROS (`http://wiki.ros.org/`). The recognized objects and their properties are further extracted as two types of representations: 1. a `Vision Graph` for referential grounding; and 2. a `Discrete State` for symbolic planning.

4.2.2.1 Vision Graph Builder

The raw recognition outcomes are firstly captured by a `Vision Graph` (computed by `Vision Graph Builder`). The `Vision Graph` captures the robot's internal representation of the shared environment. Each node in the vision graph represents a perceived object. The lower level visual properties such as color distributions, spatial locations, etc. are captured as attributes of the nodes. More details about the `Vision Graph` can be found in [65].

4.2.2.2 Discrete State Builder

The `Discrete State Builder` represents the entire environment as a conjunction of state terms, using the same representation as the initial state of a symbolic planning problem. In particular, the robot can access to the environment information (i.e., the raw object recognition outcomes) as well as its own internal status, such as the location of the gripper and whether it's open or closed. Each aspect of the state information is characterized by a state term/fluent. And a complete environment description is just the conjunction of the states for each object including the robot itself. The resulted `Discrete State` will be used later for symbolic planning. A sample `Discrete State` in the air cargo transportation domain is shown as the *Init* section in Figure 3.2.

4.2.3 Referential Grounding

The semantic representation captures key task information specified in the human language. However, without grounded to the robot's representation of perception of internal knowledge, the type of semantic is still meaningless. The goal of `Referential Grounding` is to map the entities specified in human language to objects in the robot's perception. Such that, when the human mentions "the red block on the left", the robot can identify which object in the scene is the one that human is referring to. Inspired by the graph matching algorithms widely used in computer vision community, we use the graph-based approach for referential grounding as described in [65][64]. In particular, two independent graphs are maintained through out the interaction: a `Dialogue Graph` representing entities mentioned in the language and a `Vision Graph` representing the vision information as described previously. A `Reference Resolver` applied inexact graph-matching algorithm to match the dialogue graph to the vision graph to infer what objects in its own representation of the world are referred to or described by the human during interaction. Once the references are grounded, the semantic representation becomes a `Grounded Action Frame`. For example, as shown by the `Grounded Action Frame` in Figure 4.3, "*the blue block*" refers to *B1* and "*the red block on your right*" refers to *R1*.

4.2.4 Dialogue Manager

The `Dialogue Manager` is used to decide what actions the robot should take during conversation. It is composed of three parts: a representation of the state of the dialogue; an action space for the robot; and a dialogue policy that describes which action to take under which state.

The dialogue status is computed based on the human intention a dialogue state captures (from semantic representation) and the `Grounded Action Frame`. The current space of system dialogue

acts includes asking for instructions, confirming, executing actions and updating its action knowledge base with the new acquired actions. The dialogue policy stores the (dialogue state, system act) pairs. During interaction, the `Dialogue Manager` will go through the dialogue policies, identify a dialogue state that best matches the current state, and then apply the dialogue acts associated with that state.

4.2.5 Action Modules

The `Action Modules` are used to realize a high-level action specified in the `Grounded Action Frame` with the physical arm and to learn new actions. For realizing high-level actions, if the action in the `Grounded Action Frame` has a record in the `Action Knowledge`, which keeps track of the system knowledge about various action verbs, the `Discrete Planner` will do planning to find a sequence of primitive actions to achieve the high-level action. Then these primitive actions will sequentially go through `Continuous Planner` and be translated to the trajectories of arm motors. By following these trajectories, the arm can perform the high-level action. For learning new actions, these modules will calculate state changes before and after applying the action on the focus object. Such changes of the state are generalized and stored as newly acquired verb knowledge in the `Action Knowledge` space.

4.2.6 Response Generator

Once a dialogue act is chosen by the `Dialogue Manager`, the `Response Generator` decides how to realize such response. Currently, the robot's response only includes speech, but in the future more non-verbal modalities such as gestures and head movements can also be utilized. For speech responses, several sentence templates are predefined which support parameter passing at the run time. And a speech synthesizer is responsible for translating the text to speech signals that will be

further sent to a speaker.

4.3 Action Learning through Dialogue

To enable the robot's action learning and execution capabilities, a three-tier action knowledge representation has been discussed in detail in the previous chapter. In this section, we first give a brief review of this structure. And then more details about the action execution and learning procedure will be introduced.

4.3.1 Action Modules

As shown in Figure 4.4, the action knowledge base is a three-level structure, which consists of `High-level action Knowledge`, `Discrete Planner` and `Continuous Planner`.

4.3.1.1 High-level action Knowledge

The high-level actions represent verbs specified in the human language commands. They are modeled as desired goal states rather than the action sequence taught by human. For example, the "*Stack(x,y)*" could be represented as "*On(x,y)*$\wedge$*G_Open*". If the human specifies a high-level action out of the action knowledge base, the dialogue manager will verbally request for instructions to learn the action.

4.3.1.2 Discrete Planner

The *Discrete Planner* is used to decompose a high-level verb command into a sequence of primitive actions the robot can directly execute. In our system, it is implemented as a *STRIPS* [2] planner. Specifically, a planning domain is defined by a 2-tuple $\mathcal{D} =< \mathcal{P}, \mathcal{O} >$ where

- $\mathcal{P}$: Set of predicates describing a domain.

- $\mathcal{O}$: Set of operators. Each is specified by a set of preconditions and effects. An operator is applicable only when its preconditions could be entailed in a state.

And a planning problem is defined as another 2-tuple $\mathcal{Q} =< \mathcal{I}, \mathcal{G} >$ where

- $\mathcal{I}$: Initial state, the starting environment of a problem.
- $\mathcal{G}$: Goal state, which should be achieved if the problem is solved.

In our system, $\mathcal{O}$ set includes *Open_Gripper*, *Close_Gripper* and 8 different kinds of *MoveTo* [74]. And the $\mathcal{P}$ set consists of two dimensions of the environment:

- *Arm States*: *G_Open/Close* (i.e., whether the gripper is open or closed), *G_Full/Empty* (i.e., whether the gripper has an object in it) and *G_At(x)* (i.e, location of the arm).
- *Object States*: *Top_Uclr/Clr(o)* (i.e., whether the block *o* has another block on its top), *In/Out_G(o)* (i.e., whether *o* is within the gripper fingers or not) and *On(o,x)* (i.e., *o* is supported by *x*).

The *I* and *G* are captured real-time during the dialogue interaction.

4.3.1.3 Continuous Planner

This lowest level planner defines three primitive actions: *open* (i.e., open gripper), *close* (i.e., close gripper) and *move* (i.e., move to the destination). Each primitive action is defined as a trajectory computing function, implemented as inverse kinematics. The outputs of these functions are sequences of control commands sent to each arm motor to keep the arm following the trajectories.

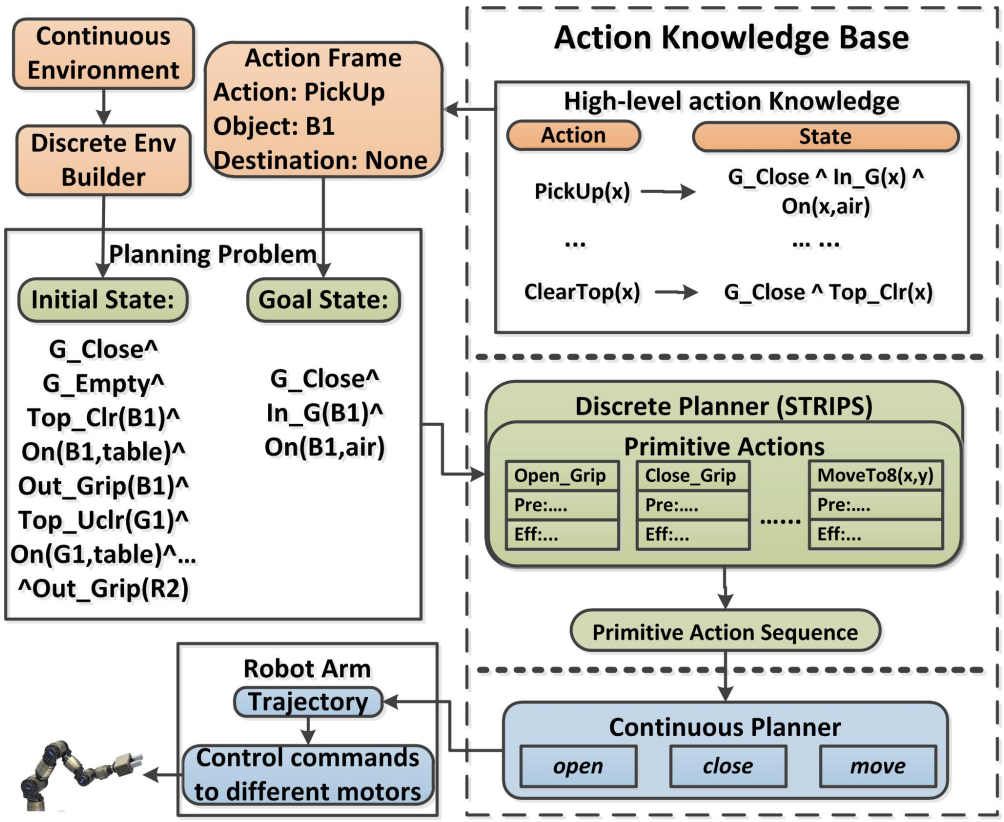


Figure 4.4 Execution example for "*Pick up the blue block*".

4.3.2 Action Execution

Given a `Grounded Action Frame`, it is firstly checked with the high-level action knowledge base. If the knowledge base has its record (e.g., the *Pickup* and *ClearTop* in Figure 4.4.), a goal state describing the action effect will be retrieved. This goal state, together with the initial state captured from the current environment, will form a planning problem and be sent to the `Discrete Planner`. And, through automated planning, the planning algorithm can generate a sequence of primitive actions to complete the task, which can be immediately executed by the arm.

Take the "*Pick up*" action frame in Figure 4.4 as an example. By checking the grounded action frame with the high-level action knowledge, a related goal state (i.e., "*G_Close* $\wedge$*In_G(B1)* $\wedge$*On(B1,air)*") can be retrieved. At the same time, the `Discrete Evn Builder` translates the raw visual information sensed from real world environment (i.e., in the form of numerical values) as a

conjunction of state literals/terms, which serves as the initial state. Given the combination of initial state and goal state, the symbolic planner can search for a path of primitive actions to solve the problem. For example, the *PickUp(B1)* in Figure 4.4 can be solved by *Open_Grip*, *MoveTo(B1)*, *Close_Grip* and *MoveTo(air)*.

The primitive actions are executed by the continuous planner and control process in the lower robotic system. For the "*open*" and "*close*", they are executed by controlling the position of the gripper fingers. For the "*move*", a task-space trajectory is first planned based on the minimum-time motion planning algorithm to move the robot end-effector from the current position to the final position. A kinematic controller with redundancy resolution [75] is then used to generate the joint movements for the robot to track the planned trajectory. Achieving the end of the trajectory indicates the action completion.

4.3.3 Action Acquisition

In our action knowledge representation, the middle level and the low level are domain knowledge, which are pre-defined in the system. The high level actions are specified by humans, which are more flexible and have more possibilities. So, the *Action Acquisition* means learning how to represent the human specified verbs as high level action knowledge.

For a new unknown action to the robot, the human will illustrate how to perform this verb action step-by-step in a learning instance. And then, by comparing the states before and after the illustration, the robot can analysis the state changes and find a set of final state terms to represent the meaning of this unknown verb. Specifically, suppose there is an unknown action frame $\mathcal{A}_i(c_1...c_k)$ specified by human utterance. $\mathcal{A}_i$ is the name of unknown action, and $c_1...c_k$ are the arguments which are objects in the scene. At the beginning of the instruction, the state of $c_1...c_k$ is a conjunction of grounded state terms that are true in the beginning environment (i.e.,

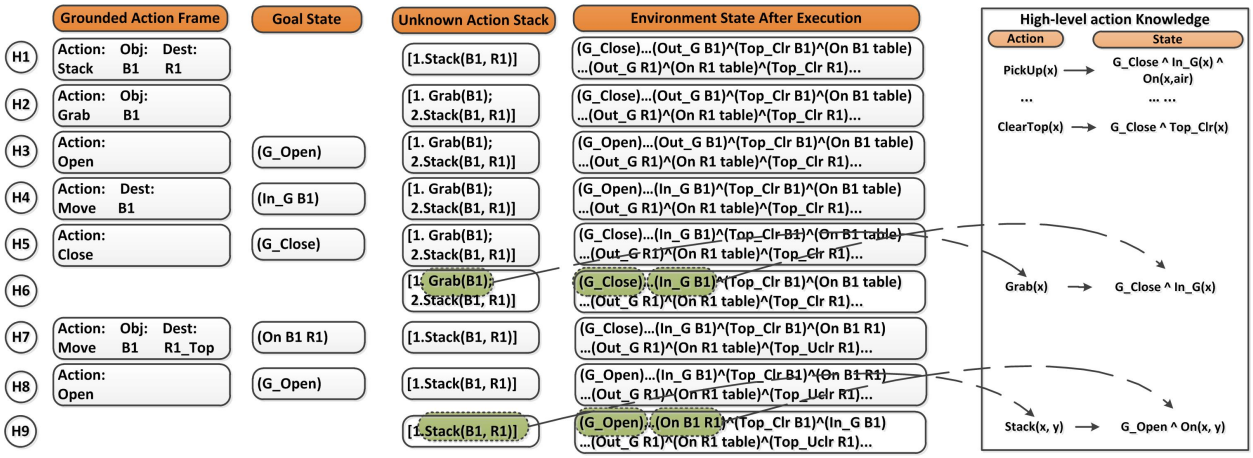


Figure 4.5 Learning process illustration. After hearing the stack action, the robot cannot perform. So the human gives step by step instruction. When the instruction is completed, new knowledge of *Grab(x)* and *Stack(x,y)* are learned in the high-level action knowledge base as the combination of the goal state of the robotic arm and the changes of the state for the involved objects. The first column shows the grounded verb frame extracted from human utterances (the corresponding dialogue is shown in Figure 4.1). If certain verb is known to the robot, the corresponding goal state will be retrieved from the knowledge space which is shown in the second column (e.g., the H_3, H_4 and etc.). The third column keeps track of the unknown verbs waiting to be learned in the form of a stack. The fourth column shows the discretized environment (i.e., in the form of conjunctions of state terms) right before the corresponding language command. And the fifth column represents the system knowledge of high-level verbs.

$\mathcal{S}_b(c_1...c_k) = p_{b1}(c_1...c_k) \wedge ... \wedge p_{bn}(c_1...c_k)$, where each p_{bj} is a fluent whose arguments include all the objects $c_1...c_k$). And at the end of the instruction, we calculate another object states $\mathcal{S}_e(c_1...c_k) = p_{e1}(c_1...c_k) \wedge ... \wedge p_{en}(c_1...c_k)$. This $\mathcal{S}_e$ represents the end states of argument objects. Then, the high level representation of $\mathcal{A}_i(c_1...c_k)$ is calculated as:

$$\mathcal{A}_i(c_1...c_k) = (\mathcal{S}_e - \mathcal{S}_e \cap \mathcal{S}_b) \cup p_{grip} \tag{4.1}$$

where $p_{grip} \in \{G\_Open, G\_Close\}$ depends on the arm status at the end of instruction.

Figure 4.5 illustrates an example learning process of acquiring novel high-level action knowledge corresponding to the dialogue in Figure 4.1.

At the beginning of the dialogue, the grounded action frame *Stack(B1, R1)* captured from the

first human utterance (i.e., H_1)is not in the action knowledge, so it will be pushed to the top of the unknown action stack as a new action waiting to be learned. The environment state at this point is calculated as shown in the figure. Then the robot will verbally request instructions. During the instruction, it's possible that another unknown action *Grab(B1)* is referred (i.e., H_2). The same as the *Stack* action, it will be pushed to the top of unknown action stack waiting to be learned.

In the next instruction, the human says "*Open your gripper*" (i.e., H_3). This sentence can be translated as action frame *Open* and the goal state "*G_Open*" can be retrieved from the action knowledge base. After executing the action sequence, the gripper state will be changed from "*G_Close*" to "*G_Open*", as shown in Figure 4.5. In the following two instructions, the human says "*Move to the blue block*" (i.e., H_4)and "*Close gripper*" (i.e., H_5). Similarly, these two instructions are translated as action frames *Move(B1)* and *Close*, then are executed accordingly. After executing these two steps, the state of *B1* is changed from "*Out_G(B1)*" to "*In_G(B1)*".

At this point, the previous unknown action *Grab(B1)* is achieved, so the human says "*Now you achieve the grab action*" (i.e., H_6) as a signal of teaching completion. After acknowledging the teaching completion, the action learning module will learn the new action representation by combining the arm state with the state changes of the argument objects in the unknown action frame. For example, the argument object of unknown action *Grab(B1)* is *B1*. By comparing the original state of *B1*, [*(Out_G B1)∧(Top_Clr B1)∧(On B1 table)*] with the final state, [*(In_G B1)∧(Top_Clr B1)∧(On B1 table)*], *B1* is changed from *(Out_G B1)* to *(In_G B1)*. So, the learning module will generalize such state changes and acquire the knowledge representation of the new action *Grab(x)* as *G_Close∧In_G(x)*. This new verb knowledge will be stored in the high-level action knowledge base and used for future interactions. The same process learns the representation of *Stack(x,y)* as $G\_Open \wedge On(x, y)$ as shown in the last sentence H_9.

4.4 Demonstration with a Physical Robot

The proposed action knowledge representation and dialogue system are implemented on a SCHUNK robotic arm. A short demo [ii] is created to establish the intelligent agent's verb acquisition capability, as well as the ability of applying the acquired new verb in novel situations. The demo setup simulates a classical AI planning problem (i.e., Blocks World [66]), but we focus on the learning capability which is not considered from the traditional problem. In this demo, similar to Figure 4.1, a set of blocks with different colors and shapes are placed on a table. The goal of the SCHUNK arm is to listen to the human commands and perform actions correspondingly. The arm can understand some of the language commands (e.g., "*move the green block on the left to the top of the green block on the right*", "*open gripper*"), but some commands may be unknown to the robots (e.g., "*sort the blocks by color*"). The human needs to teach the arm how to perform these unknown actions. To demonstrate the learning result, the acquired unknown actions are tested in new settings, after the teaching phrase.

A screen shot of the demo [ii] is shown in Figure 4.6. The interface includes an image section (i.e., displaying the perceived environment from the view point of the human and the robot) and an information section (i.e., showing the robot's intermediate understanding about the dialogue). Specifically, the information section displays following content:

1. Dialogue History: records the history of every dialogue turn between the human and the robot. For each utterance, the speech action (e.g., Confirm, IssueCommand, AskForDemo) is recorded. If the human issues a command that can directly executed by a robot, the corresponding action goal state is also captured, as shown in the Figure 4.6.

2. Robot Action Sequence: When the robot understands a human command, the robot's sym-

[ii] https://youtu.be/MGA6aqKGM0w

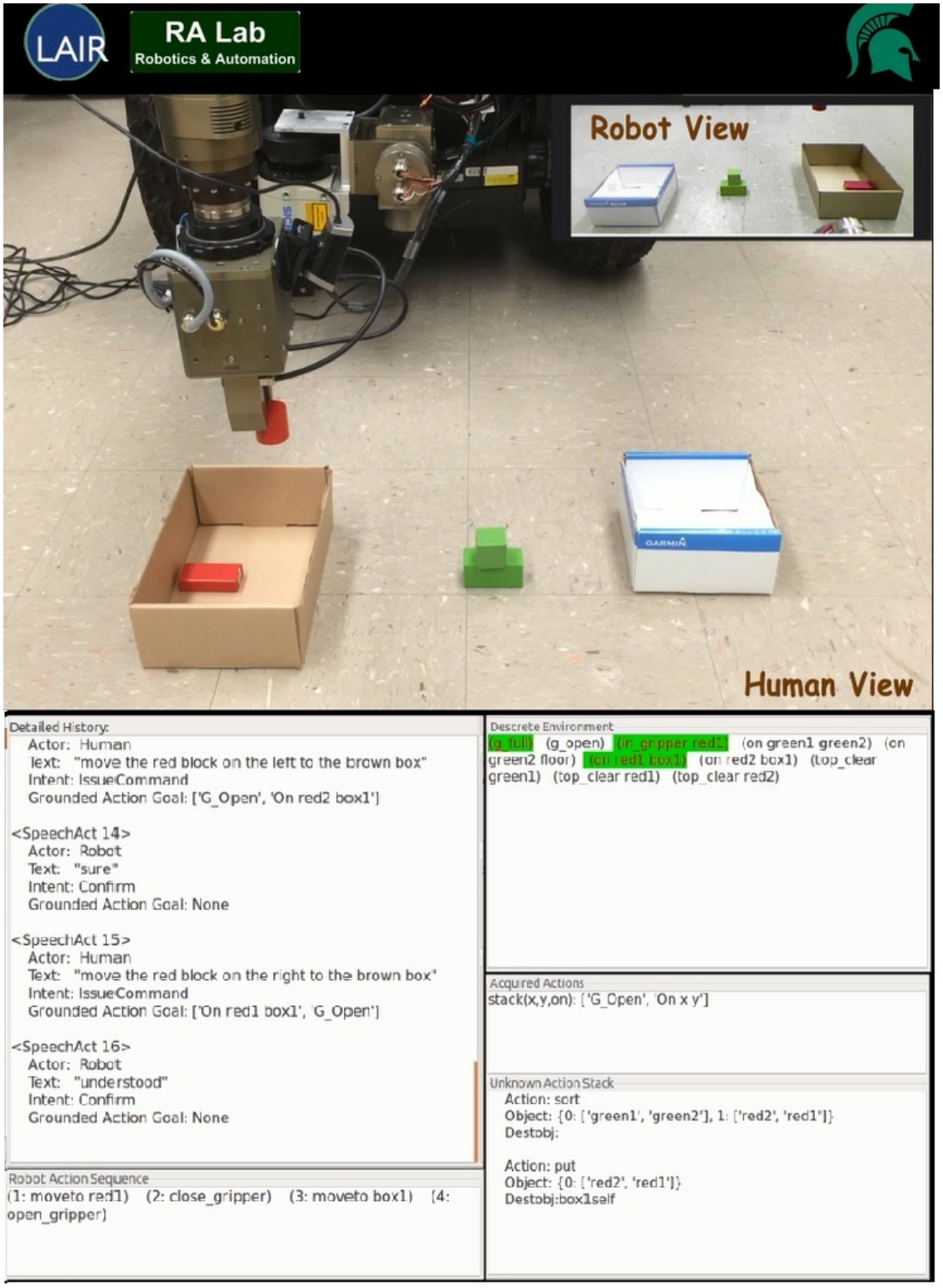


Figure 4.6 A screenshot of action learning from dialogue demo.

bolic planner can generate a sequence of primitive actions. This primitive action sequence describes exactly how would the robot perform this task step by step.

3. Discrete Environment: describes the robot's understanding of the environment in the form of a conjunction of state fluents. This vision information is updated whenever human gives an utterance. A fluent is highlighted with green background if it is changed between two consecutive perceived environments.

4. Acquired Action: shows the action knowledge acquired through current dialogue.

5. Unknown Action Stack: records the actions that human has ever mentioned and the robot does not know how to perform. Currently, this information is organized in a stack, assuming the most recent unknown action should be taught first. However, this assumption may not hold when working with end-users. In the future, more systematic algorithm should be used to help identify which unknown action the robot is learning, as well as which action has been successfully taught.

4.5 Empirical Studies

The objectives of our empirical studies are two folds. First, we aim to exam whether integrated system can support learning novel verbs from dialogue and execute the learned actions in novel situations. Second, we aim to evaluate how extra effort from the human partner through step-by-step instructions may affect the robot's learning performance.

4.5.1 Instruction Effort

Previous work on mediating perceptual differences between humans and robots have shown that a high collaborative effort from the robot leads to better referential grounding [63]. Motivated by this

previous work, we are interested in examining how different levels of effort from human partners may affect the robot's learning performance. More specifically, we model two levels of variations:

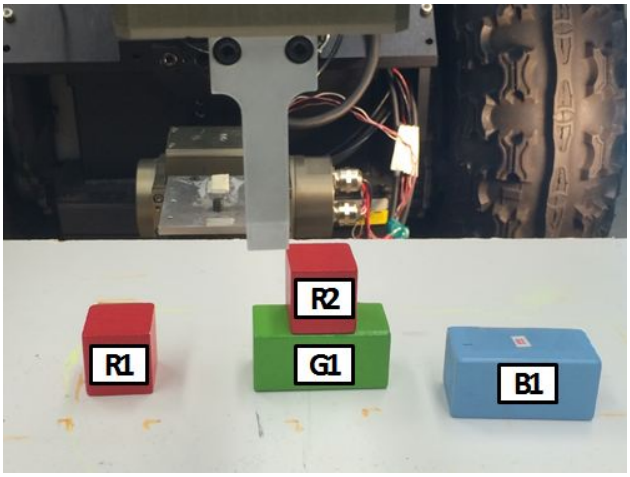
- **Collaborative Interaction:** In this setting, a human partner provides step-by-step instructions. At each step, the human will observe the the robot's response (i.e., arm movement) before moving to the next step. For example, to teach "stack", the human would issue "pick up the blue block" [iii], observe the robot's movement, then issue "put it on the red block" and observe the robot movement. By this fashion, the human makes extra effort to make sure the robot follows every step correctly before moving on. The human partner can detect potential problems and respond to immediate feedback from the robot.

- **Non-Collaborative Interaction:** In this setting, the human only provides a one-shot instruction. For example, to teach "stack", the human first issues a complete instruction "pick up the blue block and put it on top of the red block" and then observes the robot's responses. Compared to the collaborative setting, the non-collaborative setting is potentially more efficient.

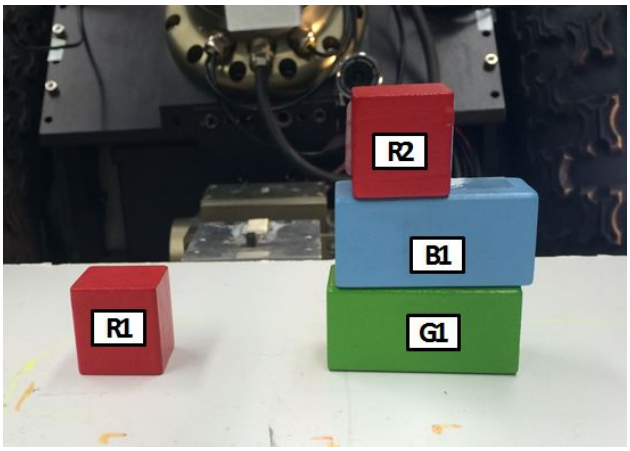
4.5.2 Experimental Tasks

Similar to the setup shown in Figure 4.1, in the study, we have multiple blocks with different colors and sizes placed on a flat surface, with a SCHUNK arm positioned on one side of the surface and the human subject seated on the opposite side. The video stream of the environment is sent to the vision system [73]. With the pre-trained object model of each block, the vision system could capture blocks' 3D positions from each frame. Five human subjects participated in our experiments. During the study, each subject was informed about the basic actions the robot can perform (i.e.,

[iii] A Sphinx speech recognizer is integrated in the system. But the speech recognition performance cannot support reliable real-time human-robot interaction. As speech recognition is not our focus, instead of using ASR results, currently the typed-in sentences are used in our experiments.



(a) Learning: the human teaches the robot how to "*pick up the blue block (i.e., B1)*" during the learning phase



(b) Execution: the human asks the robot to "*pick up the green block (i.e., G1)*" after the robot acquires the knowledge about "pick up"

Figure 4.7 Examples of a learning and an execution setup.

open gripper, close gripper, and move to) and was instructed to teach the robot several new actions through dialogue. Each subject would go through the following two phases:

4.5.2.1 Teaching/Learning Phase

Each subject was asked to teach the following five new actions under the two strategies (i.e., step-by-step instructions vs. one-shot instructions): {Pickup, Grab, Drop, ClearTop, Stack} Each time, the subject can choose any blocks they think are useful to teach the action. After finishing teaching one

action (either under step-by-step instructions or under one-shot instructions), we would survey the subject whether he/she thinks the teaching is completed and the corresponding action is successfully performed by the robot. We record the teaching duration and then re-arrange the table top setting to move to the next action.

For the teaching/learning phase, we use two metrics for evaluation: 1) *Teaching Completion Rate*(R_t) which stands for the number of actions successfully taught and performed by the robot; 2)*Teaching Completion Duration* (D_t) which measures the amount of time taken to teach an action.

4.5.2.2 Execution Phase

The goal of learning is to be able to apply the learned knowledge in novel situations. To evaluate such capability, for each action, we designed 10 additional setups of the environment which are different from the environment where the action was learned. For example, as illustrated in Figure 4.7, the human teaches the *pick Up* action by instructing the robot how to perform "*pick up the blue block(i.e., B1)*" under the environment in 4.7a. Once the knowledge is acquired about the action "pick up", we will test the acquired knowledge in a novel situation by instructing the robot to execute "*pick up the green block(i.e., G1)*" in the environment shown in 4.7b.

For the execution phase, we also used two factors to evaluate: 1) *Action Sequence Generation*(R_g) which measures how many high-level actions among the 10 execution scenarios where the corresponding lower-level action sequences are correctly generated; 2) *Action Sequence Execution*(R_{ge}) which measures the number of high level actions that are correctly executed based on the lower level action sequences.

Action learning successful rate	User 1		User 2		User 3		User 4		User 5	
	Non	Col	Non	Col	Non	Col	Non	Col	Non	Col
Pickup	1	1	1	1	1	1	1	1	0	1
Grab	1	1	1	1	1	1	1	1	1	1
Drop	1	1	1	1	1	1	1	1	0	1
ClearTop	1	1	1	1	1	1	1	1	1	1
Stack	1	1	0	1	0	1	0	1	1	1
Total	5	5	4	5	4	5	4	5	3	5

Figure 4.8 The teaching completion result of the 50 teaching dialogues. “1” stands for the dialogue where the subject considers the teaching/learning as complete since the robot performs the corresponding action correctly; and “0” indicates a failure in learning. The total numbers of teaching completion are listed in the bottom row.

4.5.3 Empirical Results

Our experiments resulted in a total of 50 action teaching dialogues. Half of these are under the step-by-step instructions (i.e., collaborative interaction) and half are under one-shot instructions (i.e., non-collaborative). As shown in Figure 4.8, 5 out of the 50 teaching dialogues were considered as incomplete by the human subjects and all of them are from the *Non-Collaborative* setting. For each of the 45 successful dialogues, an action would be learned and acquired. For each of these acquired actions, we further tested its execution under 10 different setups.

4.5.3.1 Teaching Performance

The result of teaching completion is shown in Figure 4.8. Each subject contributes two columns: the “*Non*" stands for the *Non-Collaborative* strategy and the “*Col*" column refers to the *Collaborative* strategy.

As the table shows, all the 5 uncompleted teaching are from the *Non-Collaborative* strategy. In most of these 5 cases, the subjects thought the actual performed actions were different from their expectations. For example, in one of the “*stack*" failures, the human one-shot instruction was “*move*

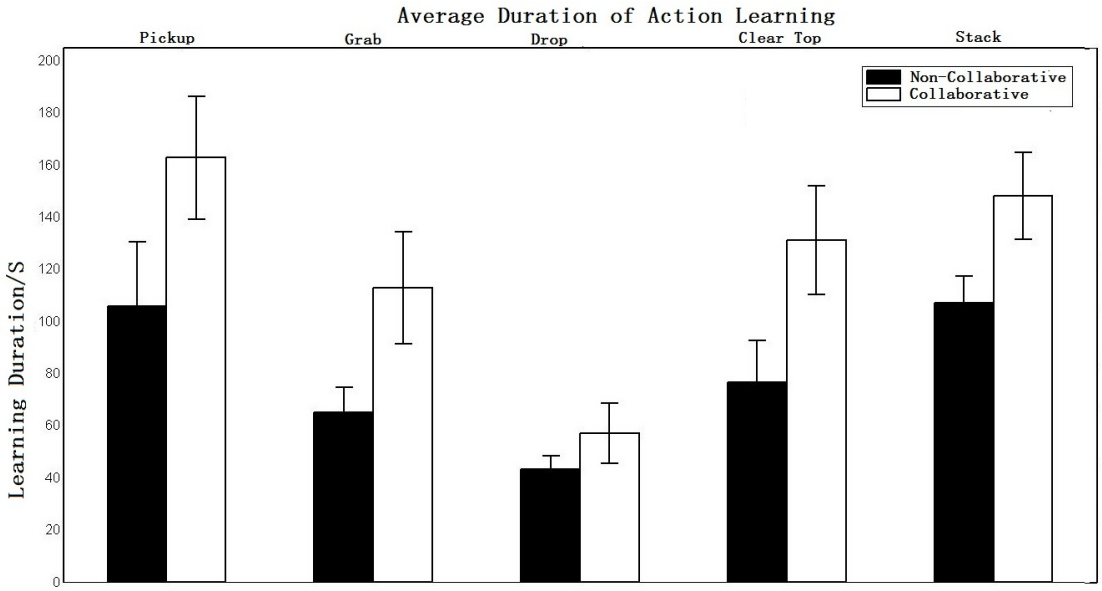


Figure 4.9 The teaching completion duration results. The durations under the non-collaborative strategy are smaller than the collaborative strategy in most cases.

the blue block to the red block on the left.". She thought the arm would put the blue block on the top of red block, open gripper and then move away. However, based on the robot's knowledge, it just moved the blue block above the red block and stopped there. So the subject considered this teaching as incomplete. On the other hand, in the *Collaborative* interactions, the robot's actual actions could also be different from the subject's expectation. But, as the instruction was given step-by-step, the instructors could notice the difference from the immediate feedback and adjust their follow-up steps, which contributed to a higher completion rate.

The duration of each teaching task is shown in Figure 4.9. Bar heights represent average teaching duration, the ranges stand for standard error of the mean (SEM). The 5 actions are represented by different groups. As shown in the figure, the teaching duration under the *Collaborative* strategy tends to take more time. Because in the *Collaborative* case, the human needs to plan next step after observing the robot's response to a previous step. If an exception happens, a sub-dialogue is often arranged to do correction. But in the *Non-Collaborative* case, the human comes up with an entire

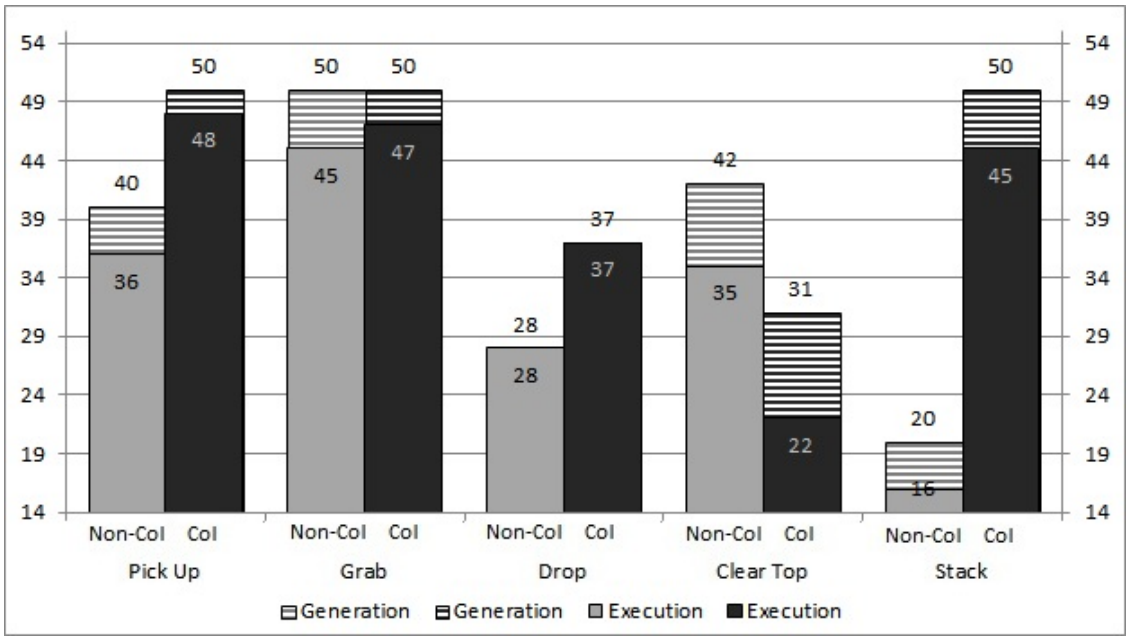


Figure 4.10 Each bar represents the number of successfully generated action sequences during testing. The solid portion of each bar represents the number of successfully executed action sequences. The number of successfully execution is always smaller than or equal to the generation. This is because we are dealing with dynamic environment, and the inaccurate real-time localization will make some correct action sequence fail to be executed.

instruction at the beginning, which appears more efficient.

4.5.3.2 Execution Performance

Figure 4.10 illustrates the action sequence generation and execution results in the execution phase.

As shown in Figure 4.10, testing results of actions learned under the Collaborative strategy are higher than the ones using *Non-Collaborative*, this is because teaching under the *Collaborative* strategy is more likely to be successful. One exception is the *Clear Top* action, which has lower generation rate under the *Col* setting. By examining the collected data, we noticed that our system failed to learn the knowledge of *Clear Top* in one of the 5 teaching phases using *Col* setting, although the human subject labeled it as successful. Another phenomenon shown in Figure 4.10 is that the generation results are always larger than or equal with the corresponding execution results. This

is caused by inaccurate localization and camera calibration, which introduced exceptions during executing the action sequence.

4.5.3.3 Examples of acquired verb knowledge

The learning results for each verbs are stored in the *High-level action Knowledge* in the form of mappings between verb frame and the corresponding goal states. Examples of acquired verb goal state representations could be found in Table 4.1. As the results shown, all the five verbs could be represented by a conjunction of two to three state terms. However, as shown in the later chapters, when the domains and the verb actions are complicated the acquired state representations may consist of more terms.

$$
\begin{aligned}
PickUp(x) &\rightarrow G\_Close \wedge In\_G(x) \wedge On(x, air) \\
ClearTop(x) &\rightarrow G\_Close \wedge Top\_Clear(x) \\
Grab(x) &\rightarrow G\_Close \wedge In\_G(x) \\
Drop(x) &\rightarrow G\_Open \wedge On(x, table) \\
Stack(x, y) &\rightarrow G\_Open \wedge On(x, y)
\end{aligned}
$$

Table 4.1 Example learning results of the five specified actions.

4.6 Conclusion

In this chapter, we describe an approach to robot action learning in a simplified blocks world. The simplifications of the environment and the tasks allow us to explore connections between symbolic representations of natural language and continuous sensorimotor representations of the robot which can support automated planning for novel situations. A dialogue system is implemented into a SCHUNK robotic arm to learn to perform new actions through human-robot dialogue in

this blocks world. In particular, the previously proposed three-tier action knowledge representation is implemented to represent system knowledge about high-level actions/verb commands. For a novel complex action, the human can give an illustration through dialogue using robot's existing action knowledge. In the end, the robot can analysis this illustration and extract a goal state for this novel action by comparing the environment changes before and after the action illustration. And this newly acquired action, represented by a goal state, can be immediately used to support further interaction. Our empirical studies have shown that, through the dialogue system and the three-tier action representation the robot was able to learn and execute basic actions in the blocks world. Furthermore, while investigating the effect of different styles of teaching procedures, the empirical results show that step-by-step instructions lead to better learning performance compared to one-shot instructions.

Chapter 5

HYPOTHESIS SPACE FOR VERB SEMANTICS

5.1 Introduction

Towards the goal of making a cognitive robot that can follow human commands and to learn new actions from dialogues with humans, previous chapters have proposed a goal state based verb representation, which can link higher-level concepts expressed by human language to lower-level primitive actions the robot is familiar with. Furthermore, this goal state representation was incorporated into a dialogue system, and the system was implemented on a SCHUNK robotic arm to conduct experiments in a simplified blocks world. Despite the fact that the goal state representation works well in the simplified blocks world, as we will see in later discussions, a single goal state is insufficient to represent the semantics of a verb when the application domains are more complex (e.g., kitchen, or living room domain), which is very common in real world situations.

To address this insufficiency issue, in this chapter, we propose a goal state hypothesis space to represent the semantics of a verb. Specifically, given a human command, if there is no knowledge about the corresponding verb (i.e., no existing hypothesis space for that verb), the robot will initiate a learning process to ask human partners to teach how to accomplish this command. After the teaching, a hypothesis space of fluents for that verb frame will be automatically acquired. If there is an existing hypothesis space for the verb, the robot will select the best hypothesis that is most relevant to the current situation and plan for the sequence of lower-level actions. Based on the outcome of

A significant portion of this chapter was published in the following paper: Lanbo She and Joyce Y. Chai. Incremental acquisition of verb hypothesis space towards physical world interaction, in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, 2016, Volume 1: Long Papers, pages 108–117.

the actions (e.g., whether it has successfully executed the command), the corresponding hypothesis space will be updated. Through this fashion, a hypothesis space for each encountered verb frame is incrementally acquired and updated through continuous interactions with human partners. In this chapter, to focus our effort on representations and learning algorithms, we applied our approach in a simulated environment where a virtual robot was used to interact with humans. Using an existing benchmark [3], our approach has demonstrated a significant performance improvement compared to a previous leading approach [3].

Comparing to previous works [3, 18, 21], our approach has three unique characteristics . First, rather than a single goal state associated with a verb, our approach captures a space of hypotheses which can potentially account for a wider range of novel situations when the verb is applied. Second, given a new situation, our approach can automatically identify the best hypothesis that fits the current situation and plan for lower-level actions accordingly. Third, through incremental learning and acquisition, our approach has a potential to support life-long learning from humans.

Following sections provide more details on a formulation of the incremental learning framework, the hypothesis space representation, the induction and inference algorithms, as well as experiments and evaluation results.

5.2 An Incremental Learning Framework

An overview of our incremental learning framework is shown in Figure 5.1. Given a language command $\mathcal{L}_i$ (e.g. *"fill the cup."*) and an environment $\mathcal{E}_i$ (e.g. a simulated environment shown in Figure 5.1), the goal is to identify a sequence of lower-level robotic actions to perform the command. Similar to previous works [76, 77], the environment $\mathcal{E}_i$ is represented by a conjunction of grounded state fluents, where each fluent describes either the property of an object or relations (e.g. spatial) between objects. The language command $\mathcal{L}_i$ is first translated to an intermediate representation

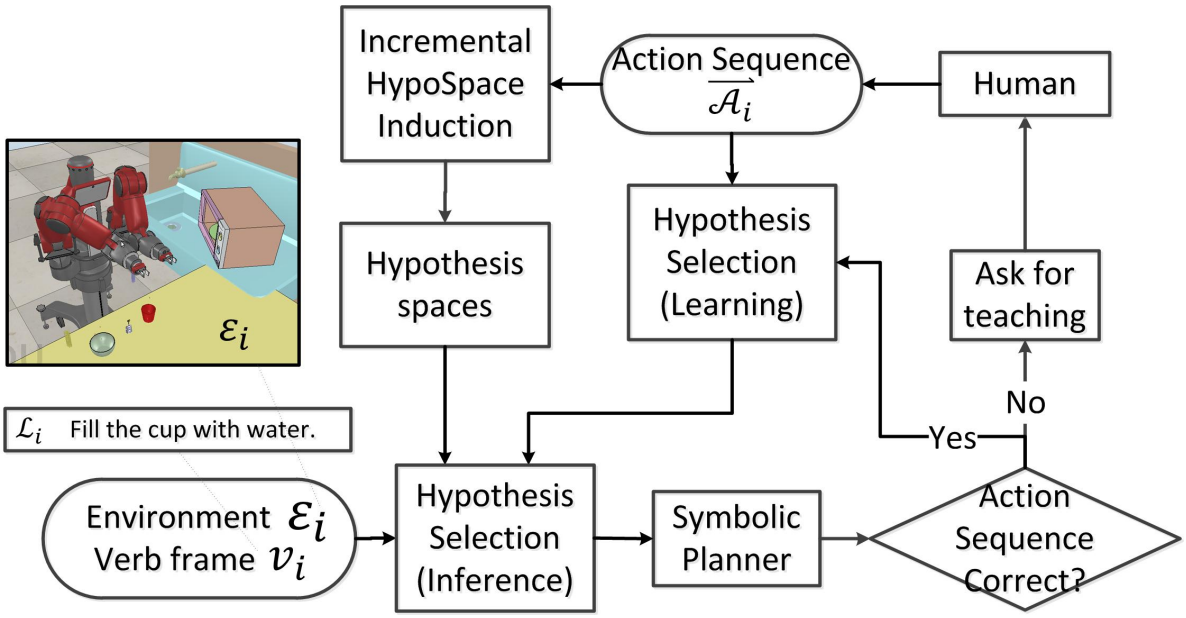


Figure 5.1 An incremental process of verb acquisition (i.e. learning) and application (i.e. inference).

of grounded verb frame v_i through semantic parsing and referential grounding[i] (e.g. For "*fill the cup*", verb "*fill*" and argument noun phrases "*cup*" are extracted and grounded to "Cup1" in the scene. Then the command is represented as $fill(Cup1)$). The system knowledge of each verb frame is represented by a *Hypothesis Space* $\mathcal{H}$, where each hypothesis (i.e. a node) is a description of possible fluents or resulting states of executing the verb command. Given a verb frame v_i and environment $\mathcal{E}_i$, a *Hypothesis Selector* will choose an optimal hypothesis from space $\mathcal{H}$ to describe the expected resulting state of executing v_i in $\mathcal{E}_i$. Given this goal state and current environment, a symbolic planner (e.g. STRIPS planner [2]) is used to generate an action sequence for the agent to execute. If the action sequence can correctly perform the command (e.g. evaluated by a human partner), the hypothesis selector can be updated with this successful prediction. On the other hand, if this action sequence is incorrect, the human partner will give a correct action sequence $\vec{A}_i$. Using $\vec{A}_i$ as the ground truth information, the system can not only update the hypothesis selector, but also

[i] Efforts have been made to address the semantic parsing and referential grounding problem [3, 14, 15]. In this work, the implementation from [3] is utilized.

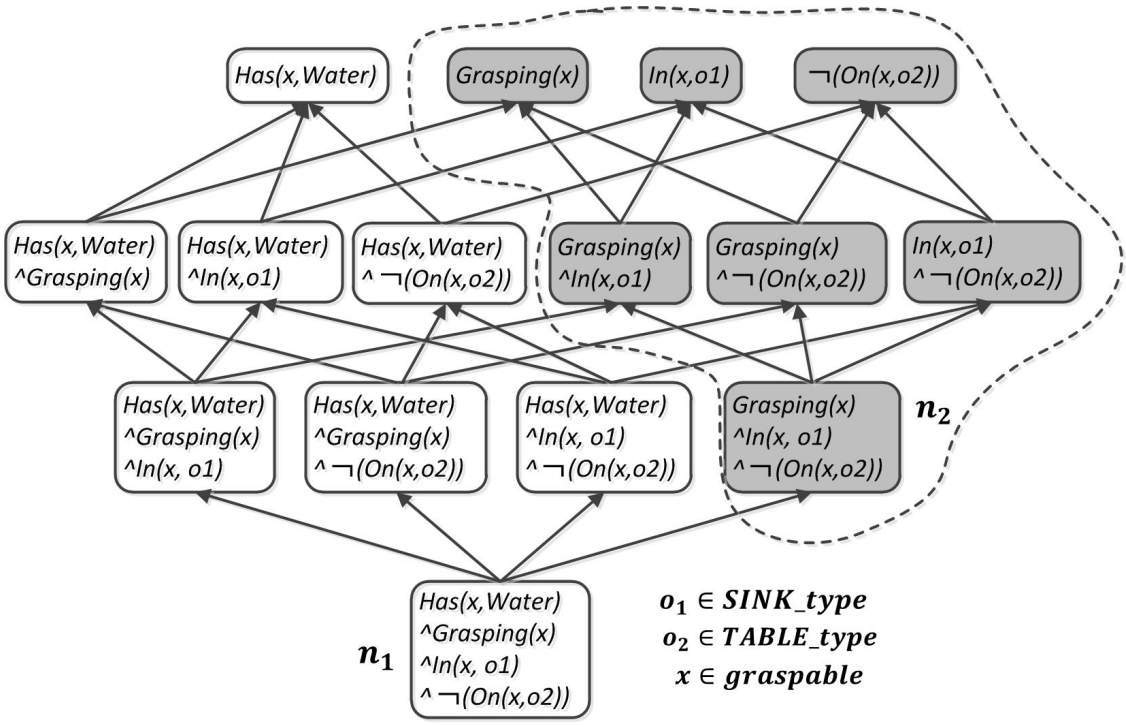


Figure 5.2 An example hypothesis space for the verb frame "*fill(x)*". The bottom node is extracted from the state changes caused by executing the "*fill*" command in an environment. Using this goal as the bottom node, the hypothesis space is generated in a bottom-up fashion, where the higher level nodes have less constrains. Each node represents a potential goal state. The highlighted nodes will be pruned during induction, as they are not consistent with the bottom node.

induce an updated hypothesis space for v_i. This induced space is treated as system knowledge to be used in future interactions.

5.3 State Hypothesis Space

To bridge the human language and robotic actions, previous works have studied representing semantic of a verb with a single resulting state [3, 21]. The major problem of this representation is that during inference if any part of the resulting state cannot be satisfied in a new situation, the symbolic planner will not be able to generate a plan. The planner is also not able to tell whether this part of state representation is even necessary. In fact, this effect is similar to the over-fitting problem. For example, in previous works, given a learning instance of performing "*fill(x)*", the

induced hypothesis could be "$Has(x, Water) \wedge Grasping(x) \wedge In(x, o_1) \wedge \neg(On(x, o_2))$", where x is a graspable object (e.g. a cup or bowl), o_1 is any type of sink, and o_2 is any table. However, during inference, when applied to a new situation which doesn't have any type of sink or table, this hypothesis will not be applicable. While the first two terms $Has(x, Water) \wedge Grasping(x)$ may be already sufficient to generate a plan to complete the task.

To handle this over-fitting problem, we propose a hierarchical hypothesis space to represent verb semantics, as shown in Figure 5.2. The space is organized based on a specific-to-general hierarchical structure. Formally, a hypothesis space $\mathcal{H}$ for a verb frame is defined as: $\langle \mathbb{N}, \mathbb{E} \rangle$, where each $n_i \in \mathbb{N}$ is a hypothesis node and each $e_{ij} \in \mathbb{E}$ is a directed edge pointing from parent n_i to child n_j meaning node n_j is more general than n_i with one less constraint.

In Figure 5.2, the bottom hypothesis (n_1) is $Has(x, Water) \wedge Grasping(x) \wedge In(x, o1) \wedge \neg(On(x, o2))$. A hypothesis n_i represents a conjunction of parameterized state fluents l_k:

$$n_i := \wedge\, l_k, and\, l_k := [\neg]\, pred_k(x_{k_1}[, x_{k_2}])$$

A fluent l_k is composed of a predicate (e.g. object status: *Has,* or spatial relation: *On*) and a set of argument variables. It can be positive or negative. Take the bottom node in Figure 5.2 as an example, it contains four fluents including one negative term (i.e. $\neg(On(x, o_2))$) and three positive terms. During inference, the parameters will be grounded to the environment to check whether this hypothesis is applicable.

5.4 Hypothesis Space Induction

Given an initial environment $\mathcal{E}_i$, a language command which contains the verb frame v_i and a corresponding action sequence $\vec{\mathcal{A}}_i$, $\{\mathcal{E}_i, v_i, \vec{\mathcal{A}}_i\}$ forms a training instance for inducing hypothesis space.

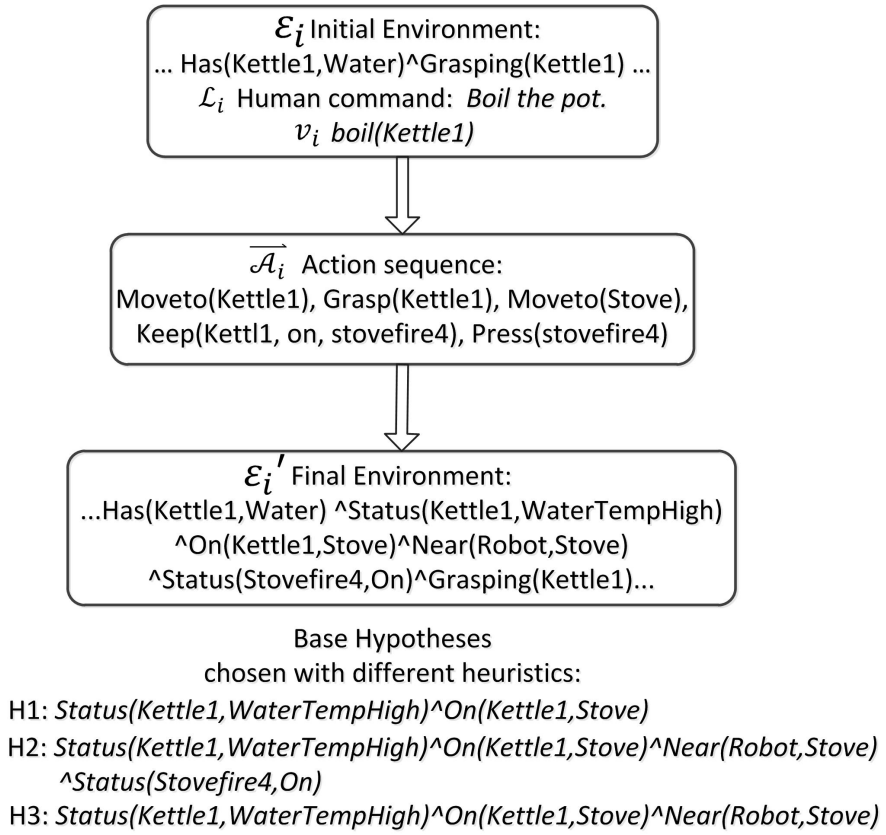


Figure 5.3 A training instance $\{\mathcal{E}_i, v_i, \vec{\mathcal{A}}_i\}$ for inducing hypothesis space. $\mathcal{E}_i'$ is the resulting environment of executing $\vec{\mathcal{A}}_i$ in $\mathcal{E}_i$. Base Hypotheses chosen with different heuristics are shown below the instance.

First, based on different heuristics, a base hypothesis is generated by comparing the state difference between the final and the initial environment. Second, a hypothesis space $\mathcal{H}$ is induced on top of this *Base Hypothesis* in a bottom-up fashion. And during induction some nodes are pruned. Third, if the system has existing knowledge for the same verb frame (i.e. an existing hypothesis space $\mathcal{H}_t$ for the same verb frame), this newly induced space will be merged with previous knowledge. Next we explain each step in detail.

5.4.1 Base Hypothesis Induction

One key concept in the space induction is the *Base Hypothesis* (e.g. the bottom node in Figure 5.2), which is the foundation of building a space. As in an example shown in Figure 5.3, given a verb

frame v_i and a working environment $\mathcal{E}_i$, the action sequence $\vec{\mathcal{A}}_i$ given by a human will change the initial environment $\mathcal{E}_i$ to a final environment $\mathcal{E}_i'$, where the state changes are highlighted in Figure 5.3. Suppose a world state change can be described by n fluents, then the first question is which of these n fluents should be included in the base hypothesis. To gain some understanding on what would be a good representation, we applied different heuristics of choosing fluents to form the base hypothesis. Example hypotheses chosen with different heuristics are shown in Figure 5.3.

- $H1_{argonly}$: only includes the changed states associated with the argument objects specified in the frame (e.g. In the example in Figure 5.3, $Kettle1$ is the only argument).

- $H2_{manip}$: includes the changed states of all the objects that have been manipulated in the action sequence taught by the human.

- $H3_{argrelated}$: includes the changed states of all the objects related to the argument objects in the final environment. An object o is considered as "related to" an argument object if there is a state fluent that includes both o and an argument object in one predicate. (e.g. the argument object $Kettle1$ is only related with $Stove$ through $On(Kettle1, Stove)$).

- $H4_{all}$: includes all the fluents whose values are changed from $\mathcal{E}_i$ to $\mathcal{E}_i'$ (e.g. all the four highlighted state fluents in $\mathcal{E}_i'$).

5.4.2 Single Space Induction

First we define the ***consistency*** between two hypotheses:

Definition. Hypotheses h_1 and h_2 are ***consistent***, if and only if the action sequence $\vec{\mathcal{A}}_1$ generated from a symbolic planner based on goal state h_1 is exactly the same as the action sequence $\vec{\mathcal{A}}_2$ generated based on goal state h_2.

The intuition behind this *consistency* is that if two goal states can lead to the same action sequence in the environment (the planner uses the same environment as initial state, but uses different goal states as the target, to generate solution action sequences), these two goal states actually have the same semantics in this environment. For example, when the planner uses "Has(x, Water)" (i.e., the top left hypothesis in Figure 5.2) as the goal state to generate an action sequence A, the generated action sequence is guaranteed to change the current environment e to a final environment e', where the specified goal "Has(x, Water)" is True in e'. One thing to notice is that this action sequence A may cause environment changes other than "Has(x, Water)" as side effects. For example, in the kitchen, when we want to fill the cup with water (i.e., using "Has(x, Water)" as the goal). The only solution is to move the cup into the sink and turn on the faucet. After these actions (i.e., move the cup into the sink and turn on the faucet), the environment changes will be: the cup is no longer in its original location (i.e., on the table or shelf), the cup is in the sink (because there's no more actions after turn on the faucet), and the cup has water inside. Even though the only "target" is Has(x, Water), the generated action sequence can also lead to "side effects" (i.e., cup is not on the table, cup is in the sink). This means that, in Figure 5.2, the top left hypothesis (i.e., "Has(x, Water)") is semantically the same as the very bottom hypothesis.

Another way to explain this is that, for those planning operators in this domain, the only operator that has "Has(x, Water)" as part of the result is the operator "turn_SinkKnob" (i.e., turn the faucet on or off depending on the current status of the faucet). And part of the pre-condition to achieve "Has(x, Water)" is to "In(x, Sink)", which is specified in the definition of operator "turn_SinkKnob". Then, consider back tracing, initially the cup is not in the sink (the pre-condition to achieve Has(x, Water) is not satisfied), the planner need to add another action Move(cup, Sink) (i.e., move the cup into the sink) before the turn sinkknob action. As a result, the entire sequence may cause more than one fluent changes where "Has(x, Water)" is only part of this entire changes. Even though "Has(x,

Input: A Base Hypothesis h
Initialization: Set initial space $\mathcal{H} : \langle \mathbb{N}, \mathbb{E} \rangle$ with $\mathbb{N}$:[h]
and $\mathbb{E}$:[ ],
Set a set of temporary hypotheses T:[h]
while T is not empty **do**
  Pop an element t from T
  Generate children [$t^{(0)}$,...,$t^{(k)}$] from t by removing each single fluent
  foreach $i = 0 \ldots k$ **do**
    if $t^{(i)}$ is consistent with t **then**
      Append $t^{(i)}$ to T;
      Add $t^{(i)}$ to $\mathbb{N}$ if not already in;
      Add link $t \rightarrow t^{(i)}$ to $\mathbb{E}$ if not already in;
    else
      Prune $t^{(i)}$ and any node that can be generalized from $t^{(i)}$
    end
  end
end
Output: Hypothesis space $\mathcal{H}$

Figure 5.4 A single hypothesis space induction algorithm. $\mathcal{H}$ is a space initialized with a base hypothesis and an empty set of links. T is a temporary container of candidate hypotheses.

Water)" was the only target for the planner, in this case, the generated action sequence lead to more fluent changes, and this entire set of changes could be the same as the base hypothesis.

The procedure to induce a hypothesis space from a single hypothesis is illustrated in Figure 5.4. Given a base hypothesis, the space induction process is a while-loop of generalizing hypotheses in a bottom-up fashion which stops when no hypotheses can be further generalized. As shown in Figure 5.4, a hypothesis node t can be generalized to a set of children hypotheses [$t^{(0)}$,...,$t^{(k)}$] by removing each single fluent from t. For example, the base hypothesis n_1 in Figure 5.2 can be generalized to 4 nodes. If a child node $t^{(i)}$ is consistent with its parent t (i.e. determined based on the *consistency* defined previously), node $t^{(i)}$ and a link $t \rightarrow t^{(i)}$ are added to the space $\mathcal{H}$. The node $t^{(i)}$ is also added to a temporary hypothesis container waiting to be further generalized. On the other hand, some children hypotheses can be inconsistent with its parent. For example, the gray node (n_2) in Figure 5.2 is inconsistent with its parent (n_1). Hypotheses that are inconsistent

with its parent are pruned. In addition, if $t^{(i)}$ is inconsistent with its parent t, any children of $t^{(i)}$ is also inconsistent with t (e.g. children of n_2 in Figure 5.2 are also gray nodes meaning they are inconsistent with the base hypothesis). After pruning, the size of entire space can be greatly reduced.

In the resulting hypothesis space, every single hypothesis is consistent with the base hypothesis. The intuition to only keep consistent hypotheses is that we want to prune out fluents that are not representative of the main goal associated with the verb.

5.4.3 Space Merging

If the robot has existing knowledge (i.e. hypothesis space $\mathcal{H}_t$) for the same verb frame, the induced hypothesis space $\mathcal{H}$ from a new instance will be merged with the existing space $\mathcal{H}_t$. Currently, a new space $\mathcal{H}_{t+1}$ is generated where the nodes of $\mathcal{H}_{t+1}$ is the union of $\mathcal{H}$ and $\mathcal{H}_t$. And links in $\mathcal{H}_{t+1}$ are generated by checking the parent-child relationship between nodes. In the future work, more space merging operations will be explored. And the human feedback in the interaction will be involved into the induction process.

5.5 Hypothesis Selection

Given a verb frame extracted from a language command, the agent will first select the best hypothesis (describing the goal state) from the existing knowledge base, and then apply a symbolic planner to generate the action sequence to achieve the goal. In our framework, the model of selecting the best hypothesis is incrementally learned throughout interactions with humans. Different models can be developed. For example, if a human can provide the feedback on whether the action sequence is correct or not, then a classifier model can be trained. In our case here, since human feedback contains the correct action sequence, we have applied a regression model.

Features on candidate hypothesis h_k and the space $\mathcal{H}_t$ 1. If h_k belongs to the top level of $\mathcal{H}_t$. 2. If h_k has the highest frequency in $\mathcal{H}_t$. **Features on h_k and current situation $\mathcal{E}_i$** 3. Portion of fluents in h_k that are already satisfied by $\mathcal{E}_i$. 4. Portion of non-argument objects in h_k. Examples of non-argument objects are o_1 and o_2 in Figure 5.2. **Similarity between testing verb frame v_i and the learning instances that induced h_k** 5. Whether the exact objects in v_i have been used in learning instances that can induce h_k. 6. The similarity of noun phrase description of v_i and the corresponding learning instances.

Table 5.1 Current features used for incremental learning of the regression model. The features are real-valued, except for the first two that are binary features.

5.5.1 Inference

Given a verb frame v_i extracted from a language command and a working environment $\mathcal{E}_i$, the goal of inference is to estimate how well each hypothesis h_k from a space $\mathcal{H}_t$ can represent the result of performing v_i in $\mathcal{E}_i$. And the best fit hypothesis will be used as the goal state to generate the action sequence. Specifically, the "wellness" of describing command v_i with hypothesis h_k in environment $\mathcal{E}_i$ is formulated as follows:

$$f(h_k \mid v_i; \mathcal{E}_i; \mathcal{H}_t) = W^T \cdot \Phi(h_k, v_i, \mathcal{E}_i, \mathcal{H}_t) \tag{5.1}$$

where $\Phi(h_k, v_i, \mathcal{E}_i, \mathcal{H}_t)$ is the feature vector and W is the weight incrementally updated during interaction[ii]. For the feature vector, a list of features is shown in Table 6.3 including multiple aspects of relations between $h_k, v_i, \mathcal{E}_i$ and $\mathcal{H}_t$. Example global features are whether the candidate goal h_k is in the top level of entire space $\mathcal{H}_t$ and whether h_k has the highest frequency. Example local features include if most of fluents in h_k are already satisfied in current scene $\mathcal{E}_i$ (as this h_k is

[ii]The SGD regressor in the scikit-learn [78] is used to perform the linear regression with l2 regularization.

unlikely to be a desired goal state). The features also include if the same verb frame v_i has been performed in a similar scene during previous interactions, as the corresponding hypotheses induced during that experience are more likely to be preferred.

5.5.2 Parameter Estimation

During interaction, when the human partner gives an action sequence $\vec{\mathcal{A}}_i$ to illustrate how to correctly perform command v_i in environment $\mathcal{E}_i$, the model weights will be incrementally updated with:

$$W_{t+1} = W_t - \eta(\alpha \frac{\partial R(W_t)}{\partial W_t} + \frac{\partial L(J_{ki}, f_{ki})}{\partial W_t})$$

where $f_{ki} := f(h_k|v_i; \mathcal{E}_i; \mathcal{H}_t)$ is defined in Equation 5.1. J_{ki} is the dependent variable the model should approximate, where $J_{ki} := J(s_i, h_k)$ is the Jaccard Index (details in Section 5.6) between hypothesis h_k and a set of changed states s_i (i.e. the changed states of executing the illustration action sequence $\vec{\mathcal{A}}_i$ in current environment). $L(J_{ki}, f_{ki})$ is a squared loss function. $\alpha R(W_t)$ is the regularization term, and η is the constant learning rate.

During training, our goal is to learn an approximation function f_{ki} (i.e., how well to use a hypothesis h_k as the goal, given the command v_i and the current environment $\mathcal{E}_i$). In the training, what we are given are: the ground truth environment changes (i.e., s_i), the command v_i, the environment $\mathcal{E}_i$, and a hypothesis space which includes a set of hypotheses h_ks. For each hypothesis h_k, the distance J_{ki} between h_k and s_i (i.e., their jaccard index) is calculated. If a J_{ki} is very low, it means given current command v_i and environment $\mathcal{E}_i$, using h_k as the goal state will not be a good choice, because this candidate h_k is very different from the ground truth goal s_i labeled by a human. Otherwise, a higher J_{ki} means h_k is a suitable choice for the command and environment. Here, value J_{ki} is the target we are trying to approximate using a function f_{ki} (the parameters in

f_{ki} are what we want to get from the training process), such that, during inference, given a new command, an environment and a candidate hypothesis, the predicator can have a "guess" about how well to choose this hypothesis as the goal. Basically, during training, the parameters in f_{ki} are the training outcomes, and J_{ki} (i.e., the jaccard index between s_i and a particular h_k) is the target for the training iterations to tune those parameters.

5.6 Experiment Setup

Dataset Description: To facilitate comparison, we adopted the dataset made available by [3]. To support incremental learning, each single utterance from a paragraph is extracted, where each command/utterance only contains one verb and its arguments. The corresponding initial environment and an action sequence taught by a human for each command are also extracted. An example is shown in Figure 5.3, where $\mathcal{L}_i$ is a language command, $\mathcal{E}_i$ is the initial working environment, and $\vec{\mathcal{A}}_i$ is a sequence of robotic actions to complete the command given by a human.

In the original data, some sentences are not aligned with any actions, which cannot be used for either learning or evaluation. After removing these no-alignment sentences, 991 data instances are collected in total, including 165 different verb frames. Some frames have higher frequency like put(x, y), take(x), fill(x, y), pour(x). Some frames may have very low frequency, like serve(x), wash(x), change(x). For these very low frequency frames, they may never appear in the training set and can lead very low testing scores. As a result, in the evaluation, we first reported an over all result (including those low frequency frames) and then reported another result only on the top four most frequent verb frames (ignoring those low frequency frames). The same verb with different numbers of arguments are considered as different verb frames. 80% of the data (793) are used for incremental space induction and learning hypothesis selector. The hypothesis spaces and regression based selectors acquired at each run are evaluated on the other 20% (198) testing instances. More

specifically, for each testing instance, the induced space and the hypothesis selector are applied to identify a desired goal state. Then a symbolic planner[iii] is applied to predict an action sequence $\vec{\mathcal{A}}^{(p)}$ based on this predicted goal state. We then compare $\vec{\mathcal{A}}^{(p)}$ with the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ using the following two metrics.

- AED: The Action sequence Edit Distance evaluating similarity d' between the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ and the predicted sequence $\vec{\mathcal{A}}^{(p)}$. Specifically, the Edit Distance d between $\vec{\mathcal{A}}^{(g)}$ and $\vec{\mathcal{A}}^{(p)}$ is first calculated. Then the d is rescaled as $d' = 1 - d/max(\vec{\mathcal{A}}^{(g)}, \vec{\mathcal{A}}^{(p)})$, such that a larger d' means the two sequences are more similar.

- SJI: Because different action sequences could lead to a same goal state, we also use Jaccard Index to check the overlap between the changed states. Specifically, executing the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ in the initial scene $\mathcal{E}_i$ results in a final environment $\mathcal{E}_i'$. Suppose the changed states between $\mathcal{E}_i$ and $\mathcal{E}_i'$ is $c^{(g)}$. For the predicted action sequence, we can calculate another set of changed states $c^{(p)}$. The Jaccard Index between $c^{(g)}$ and $c^{(p)}$ is evaluated.

Configurations: We also compare the results of using the regression model based selector with the following different strategies of selecting the hypothesis:

- *Misra2015 [3]*: The state of the art system reported in [3] on the command/utterance level evaluation[iv].

- *MemoryBased*: Given the induced space, only the base hypotheses h_ks from each learning instances are used. Because these h_ks don't have any relaxation, they represent purely learning from memorization.

[iii] The symbolic planner implemented by [69] is utilized to generate action sequences.

[iv] Here we apply the same system to predict action sequences at the command level, not at the paragraph level as reported in [3]

- *MostGeneral*: In this case, only those hypotheses from the top level of the hypothesis space are used, which have least number of state fluents. These nodes are the most relaxed hypotheses in the space.

- *MostFrequent*: In this setting, the hypotheses that are most frequently observed in the learning instances are used.

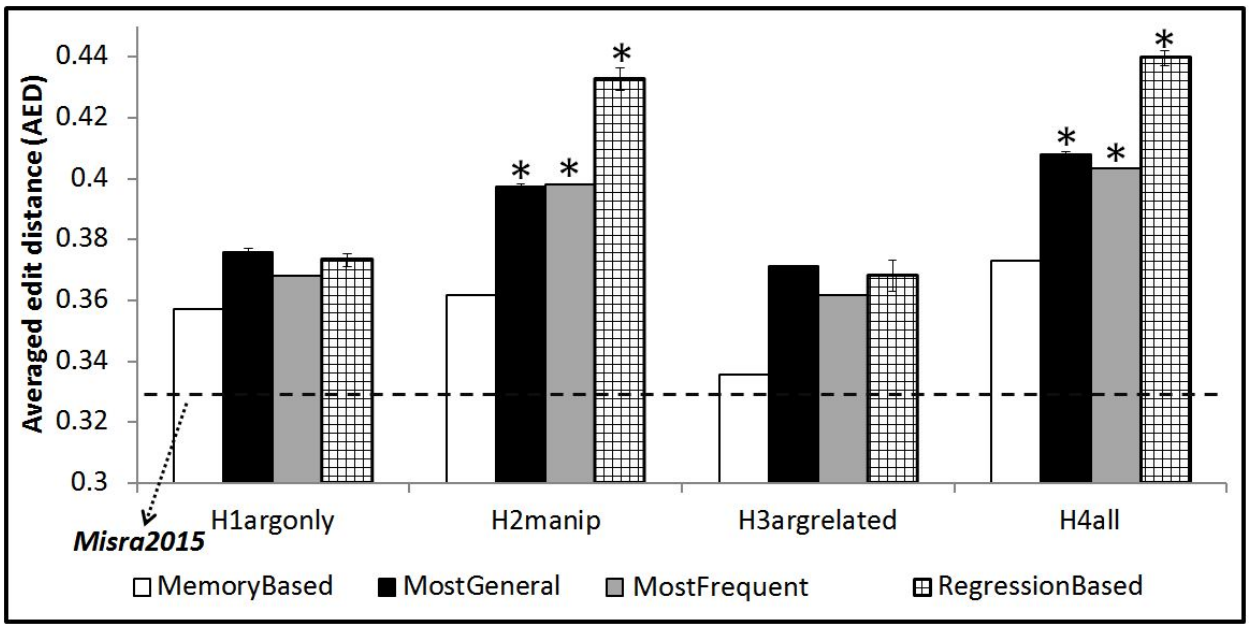
5.7 Results

5.7.1 Overall performance

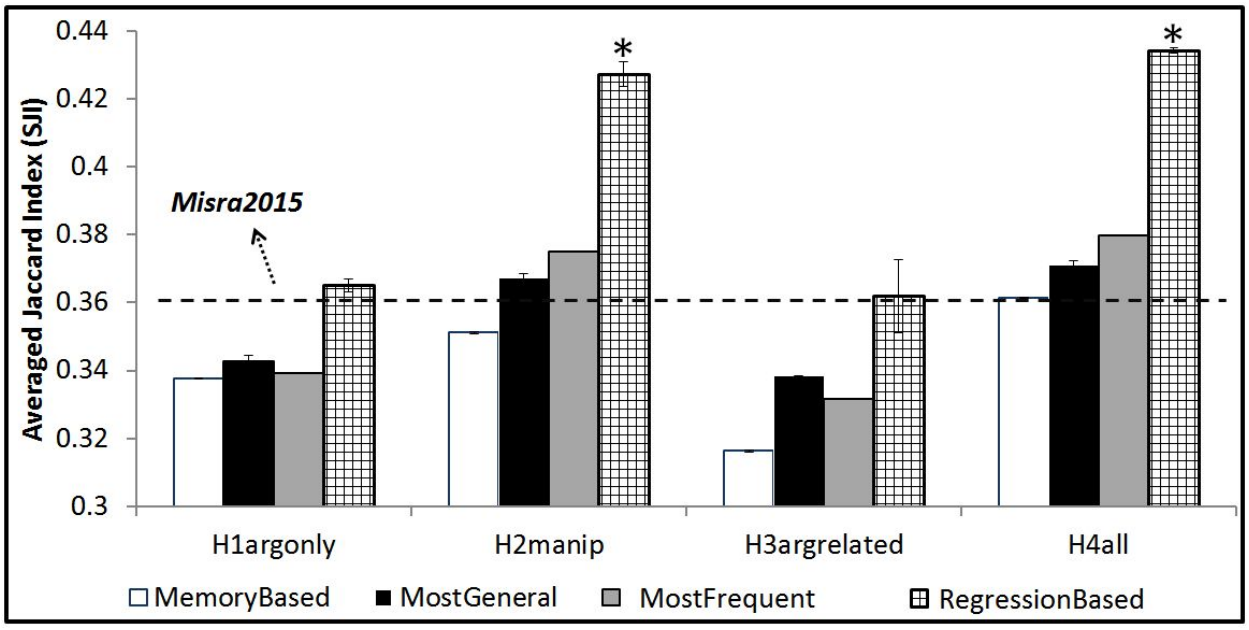
The results of the overall performance across different configurations are shown in Figure 5.5. For both of the AED and SJI (i.e. Figure 5.5a and Figure 5.5b), the hypothesis spaces with the regression model based hypothesis selector always achieve the best performance across different configurations, which outperform the previous approach [3]. For different base hypothesis induction strategies, the $H4_{all}$ considering all the changed states achieves the best performance across all configurations. This is because $H4_{all}$ keeps most of the state change information comparing with other heuristics. The performance of $H2_{manip}$ is similar to $H4_{all}$. The reason is, when all the manipulated objects are considered, the resulted set of changed states will cover most of the fluents in $H4_{all}$. On the other dimension, the regression based hypothesis selector achieves the best performance and the *MemoryBased* strategy has the lowest performance. Results for *MostGeneral* and *MostFrequent* are between the regression based selector and *MemoryBased*.

5.7.2 Incremental Learning Results

Figure 5.6 presents the incremental learning results on the testing set. To better present the results, we show the performance based on each learning cycle of 40 instances. And the averaged Jaccard

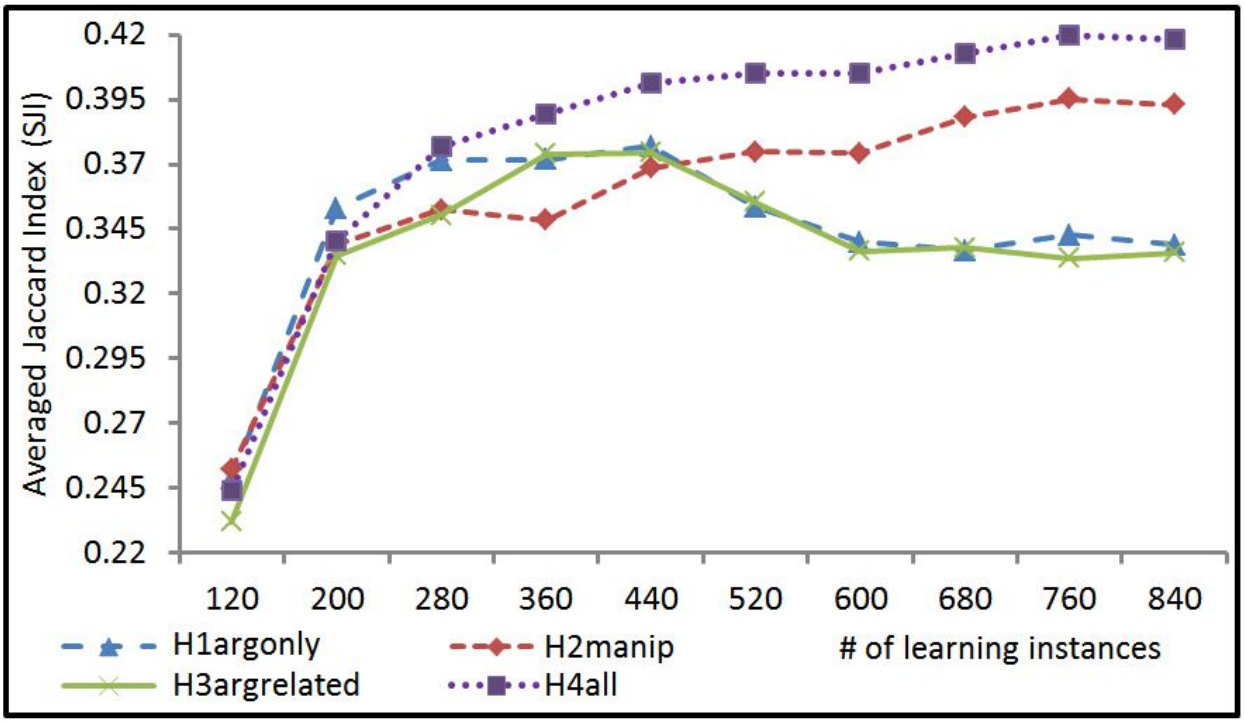


(a) AED results for different configurations

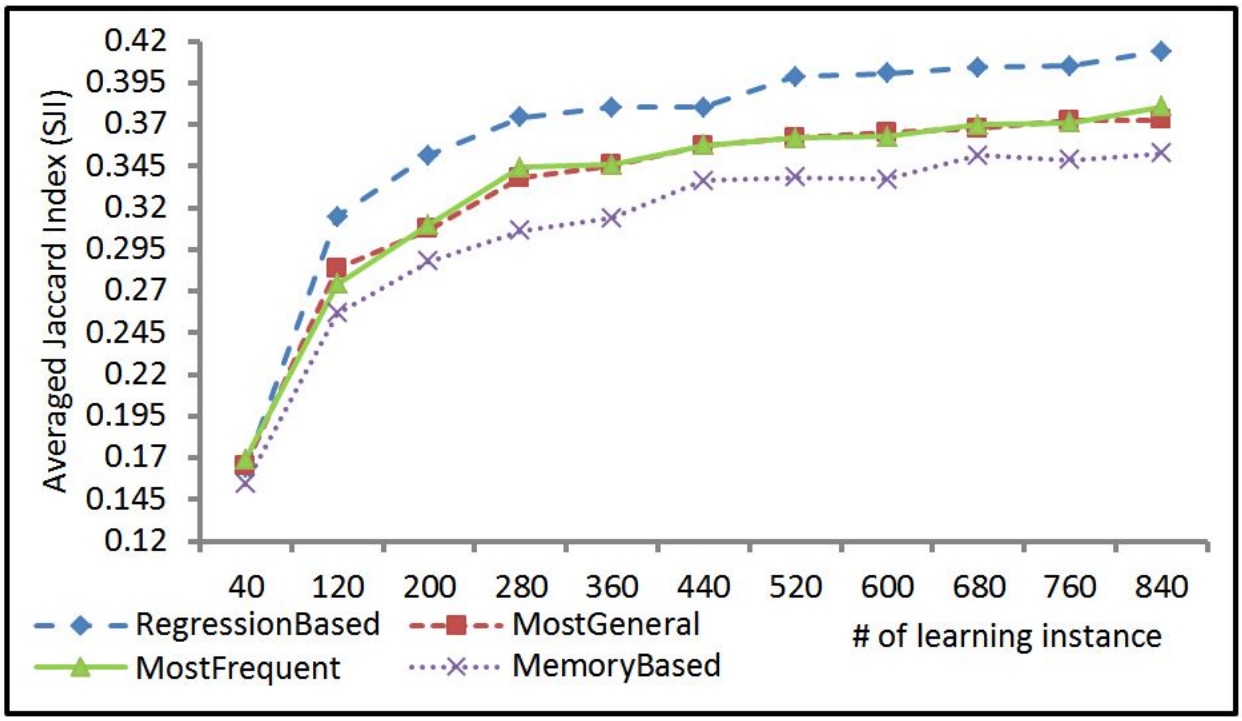


(b) SJI results for different configurations

Figure 5.5 The overall performance on the testing set with different configurations in generating the base hypothesis and in hypothesis selection. Each configuration runs five times by randomly shuffling the order of learning instances, and averaged performances are reported. The results from *Misra2015* are shown as a line. Results that are statistically significant better than *Misra2015* are marked with $*$ (*paired t-test,* $p < 0.05$).



(a) Use regression based selector to select hypothesis, and compare each base hypothesis induction heuristics.



(b) Induce the base hypothesis with $H4_{all}$, and compare different hypothesis selection strategies.

Figure 5.6 Incremental learning results. The resulting spaces and regression models are evaluated on testing set. The averaged Jaccard Index is reported.

Index (SJI) is reported. Specifically, Figure 5.6a shows the results of configurations comparing different base hypothesis induction heuristics using regression model based hypothesis selection. In the figure, after using 200 out of 840 (23.8%) learning instances, all the four curves could achieve more than 80% of the overall performance. For example, for the heuristic $H4_{all}$, the final average Jaccard Index is 0.418. And when 200 instances are used, the score is 0.340 ($0.340/0.418 \approx 81\%$). The same number holds for the other heuristics. After 200 instances, $H4_{all}$ and $H2_{manip}$ consistently achieve better performance than $H1_{argonly}$ and $H3_{argrelated}$. This result indicates that while change of states mostly affect the arguments of the verbs, other state changes in the environment cannot be ignored. Modeling them actually led to better performance. Using $H4_{all}$ for base hypothesis induction, Figure 5.6b shows the results of comparing different hypothesis selection strategies. The regression model based selector always outperform other selection strategies. And it can achieve 85% ($0.3514/0.4139 \approx 85\%$) of the final score when 25% data points are used.

5.7.3 Results on Frequent Verb Frames

Beside overall evaluation, we have also taken a closer look at individual verb frames. Most of the verb frames in the data have a very low frequency, which cannot produce significant results. So we only selected verb frames with frequency larger than 40 in this evaluation. For each verb frame, 60% data are used for incremental learning and 40% are for testing. And for each frame, a regression based selector is trained separately. The resulting SJI curves are shown in Figure 5.7.

As shown in Figure 5.7, all the four curves become steady after 8 learning instances are used. However, some verb frames have final SJIs of more than 0.55 (i.e. $take(x)$ and $turn(x)$), and some frames have relatively lower results (e.g. results for $put(x, y)$ are lower than 0.4). After examining the learning data that contributes to $put(x, y)$, we found these data are more noisy than the training data for other frames. One source of the errors is the incorrect object grounding results. For exam-

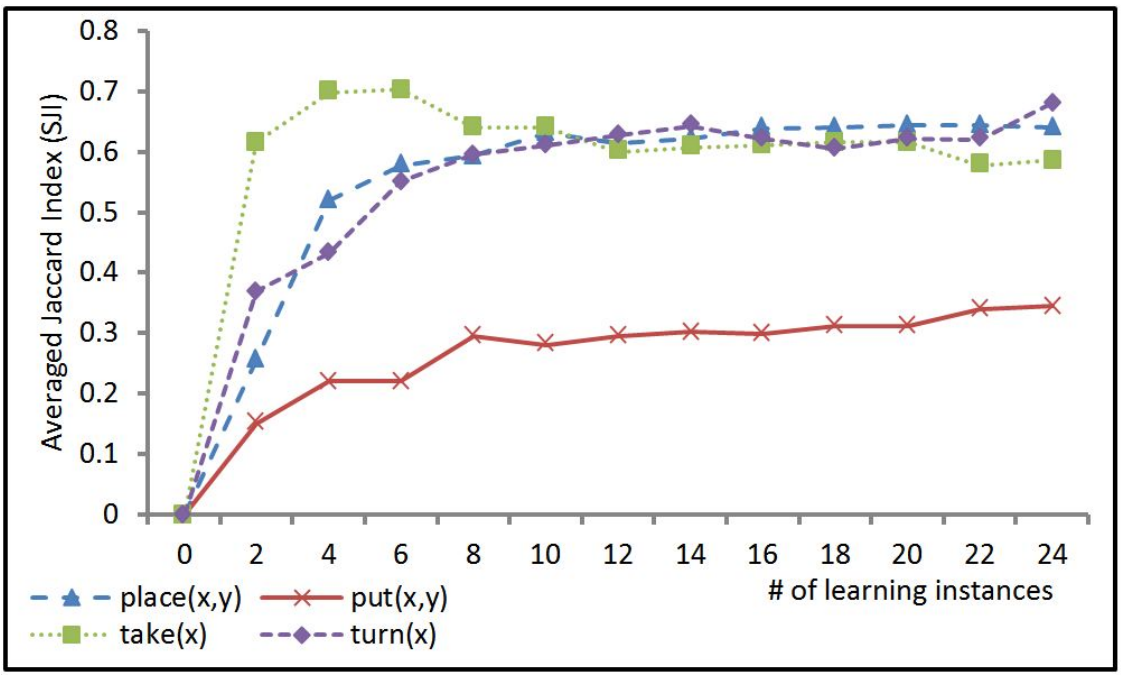


Figure 5.7 Incremental evaluation for individual verb frames. Four frequent verb frames are examined: $place(x, y)$, $put(x, y)$, $take(x)$, and $turn(x)$. X-axis is the number of incremental learning instances, and Y-axis is the averaged SJI computed with $H4_{all}$ base hypothesis induction and regression based hypothesis selector.

ple, an error training instance is "*put the pillow on the couch*", where the object grounding module cannot correctly ground the "*couch*" to the target object. As a result, the changed states of the second argument (i.e. the "*couch*") are incorrectly identified, which will mislead the prediction results during inference. Another common error source is from the utterance parsing. These errors make the nodes in the entire hypothesis space inconsistent with each other. Such that during inference, the hypothesis selection strategies fail to predict a correct goal state, even when using regression based selection. These different types of errors are difficult to be recognized by the system itself. This points to the future direction of involving humans in a dialogue to learn a more reliable hypothesis space for verb semantics.

5.8 Conclusion

In this chapter, we present an incremental learning approach that represents and acquires semantics of action verbs based on state changes of the environment. Specifically, we propose a hierarchical hypothesis space, where each node in the space describes a possible effect of the world resulted by the verb. Given a language command and a space of goal state hypotheses, the learned hypothesis selector can be applied by the agent to choose the most suitable goal state and plan for lower-level actions. Our empirical results have demonstrated a significant improvement in performance compared to a previous leading approach. More importantly, as our approach is based on incremental learning, it can be potentially integrated in a dialogue system to support life-long learning from humans.

Chapter 6

INTERACTIVE LEARNING TO ACQUIRE GROUNDED VERB SEMANTICS

6.1 Introduction

To support learning of grounded verb semantics, we have proposed goal state based verb representations in previous chapters. Following human instructions or demonstrations to execute an unknown task, robots capture the state change of the environment caused by the actions and represent verb semantics as the desired goal states. One advantage of such state-based representation is that, when robots encounter same verbs/commands in a new situation, the desired goal state will trigger the action planner to automatically plan for a sequence of primitive actions to execute the command. In chapter 5, we further extend the single goal state representation to a hypothesis space to take the various outcomes of complex tasks (e.g., *boil the water*) into consideration.

While the state-based representation of verb semantics provides an important link to connect verbs to the robot's actuator, previous approaches also present several limitations. First of all, previous approaches were developed under the assumption of perfect perception of the environment. However, this assumption does not hold in real-world situated interaction. The robot's representation of the environment is incomplete and error-prune often due to its limited sensing capabilities. Thus it is not clear whether previous approaches can scale up to handle noisy and incomplete environment.

Second, most previous works rely on multiple demonstration examples to acquire grounded verb

A significant portion of this chapter was published in the following paper: Lanbo She and Joyce Y. Chai, Interactive Learning of Grounded Verb Semantics towards Human-Robot Communication. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, 2017, pages 1634–1644.

models. Each demonstration is simply a sequence of primitive actions associated with a verb. No other type of interaction between humans and robots is explored. Previous cognitive studies [60] on how people learn have shown that social interaction (e.g., conversation with teachers) can enhance student learning experience and improve learning outcomes. For robotic learning, previous work [4] has also demonstrated the necessity of question answering in the learning process. Thus, in our view, interactive learning beyond demonstration of primitive actions should play a vital role for the robot to acquire more reliable models of grounded verb semantics. This is especially important when the perception of the world is noisy and incomplete, human language can be ambiguous, and the robot may lack relevant linguistic or world knowledge during the learning process.

Therefore, to address these limitations, we have developed a new interactive learning approach where robots actively engage with humans to acquire models of grounded verb semantics. Our approach explores the space of interactive question answering between humans and robots during the learning process. In particular, motivated by previous work on robot learning [4], we designed a set of questions that are pertinent to verb semantic representations. We further applied reinforcement learning to learn an optimal policy that guides the robot on when to ask what questions to maximize a long-term reward of learning. Our empirical results have shown that this interactive learning process leads to more reliable representations of grounded verb semantics, which contribute to significantly better action performance in new situations. When the environment is noisy and uncertain (as in the realistic situation), the models acquired from interactive learning result in a performance gain between 48% and 145% when applied in new situations. Our results further demonstrate that the interaction policy acquired from reinforcement learning leads to most efficient interaction and most reliable verb models.

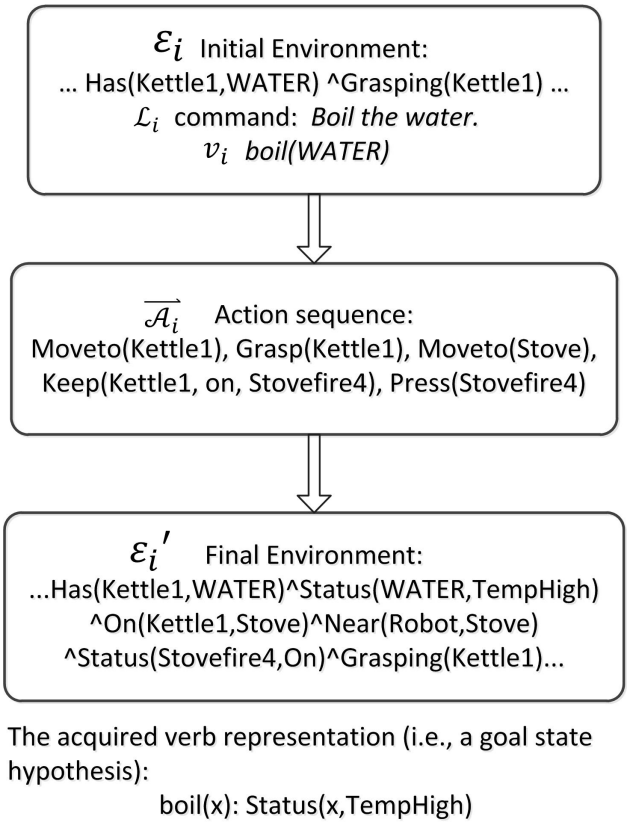


Figure 6.1 An example of acquiring state-based representation for verb semantics based on an initial environment $\mathcal{E}_i$, a language command $\mathcal{L}_i$, and a primitive action sequence $\mathcal{A}_i$demonstrated by the human, and a final environment $\mathcal{E}_i'$ resulted by executing $\mathcal{A}_i$ in $\mathcal{E}_i$.

6.2 Acquisition of Grounded Verb Semantics

This section gives a brief review on acquisition of grounded verb semantics and illustrates the differences between previous approaches and our approach using interactive learning.

6.2.1 State-based Representation

As shown in Figure 6.1, the verb semantics (e.g., *boil(*x*)*) is represented by the goal state (e.g., $Status(x, TempHigh)$) which is the result of the demonstrated primitive actions. Given the verb phrase *boil the water* (i.e., $\mathcal{L}_i$), the human teaches the robot how to accomplish the corresponding action based on a sequence of primitive actions $\overrightarrow{\mathcal{A}_i}$. By comparing the final environment $\mathcal{E}_i'$ with

the initial environment $\mathcal{E}_i$, the robot is able to identify the state change of the environment, which becomes a hypothesis of goal state to represent verb semantics. Compared to procedure-based representations, the state-based representation supports automated planning at the execution time. It is environment-independent and more generalizable. In [1], instead of one hypothesis, it maintains a specific-to-general hypothesis space as shown in Figure 6.2 to capture all goal hypotheses of a particular verb frame. Specifically, it assumes that one verb frame may lead to different outcomes under different environments, where each possible outcome is represented by one node in the hierarchical graph and each node is a conjunction of multiple atomic fluents. [i]

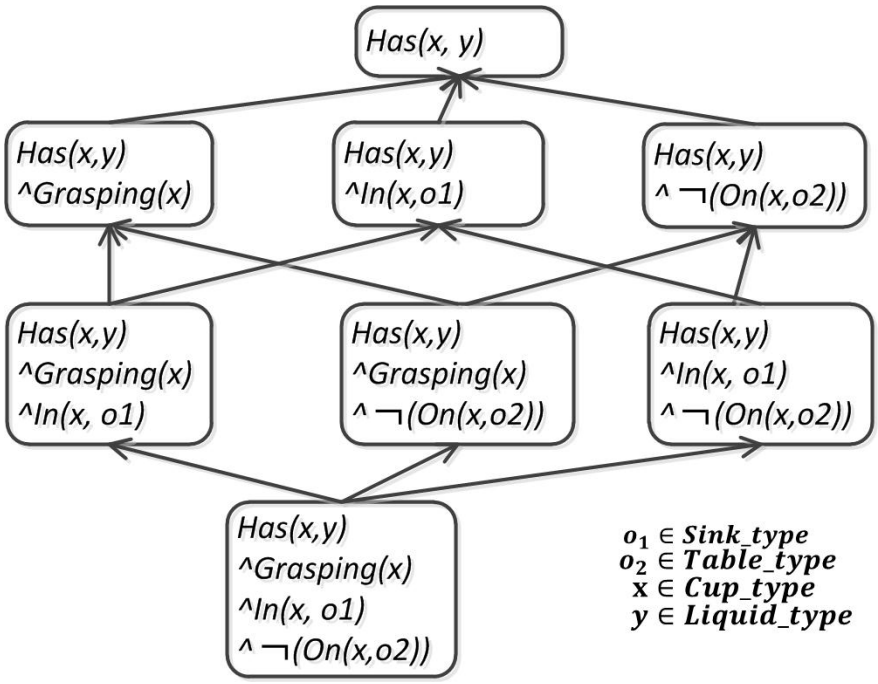


Figure 6.2 An example hypothesis space for the verb frame $fill(x, y)$.

Given a language command (i.e., a verb phrase), a robot will engage in the following processes:

- **Execution.** In this process, the robot will select a hypothesis from the space of hypotheses that is most relevant to the current situation and use the corresponding goal state to plan for

[i] In this work, we assume the set of atomic fluents representing environment state are given and do not address the question of whether these predicates are adequate to represent a domain.

actions to execute.

- **Learning.** When the robot fails to select a hypothesis or fails to execute the action, it will ask the human for a demonstration. Based on the demonstrated actions, the robot will learn a new representation (i.e., new nodes) and update the hypothesis space.

6.2.2 Noisy Environment

Probabilistic Environment:
...^Has(Kettle1,Water) 0.64 ^Grasping(Kettle1) 0.91
^Status(Kettle1,HighTemp) 0.95 ^On(Kettle1,Stove) 0.2
^Near(Robot,Stove) 0.43 ^Status(Stovefire4,On) 0.6 ^...

Figure 6.3 An example probabilistic sensing result.

Previous works represent the environment $\mathcal{E}_i$ as a conjunction of grounded state fluents. Each fluent consists of a predicate and one or more arguments (i.e., objects in the physical world, or object status), representing one aspect of the perceived environment. An example of a fluent is "$Has(Kettle_1, WATER)$" meaning object $Kettle_1$ has some water inside, where Has is the predicate, and $Kettle_1$ and $WATER$ are arguments. The set of fluents include the status of the robot (e.g., $Grasping(Kettle_1)$), the status of different objects (e.g., $Status(WATER, TempHigh)$), and relations between objects (e.g., $On(Kettle_1, Stove)$). One limitation of the previous works is that the environment has a perfect, deterministic representation, as shown in Figure 6.1. This is clearly not the case in the realistic physical world.

In reality, given limitations of sensor capabilities, the environment representation is often partial, error prone, and full of uncertainties. Figure 6.3 shows an example of a more realistic representation where each fluent comes with a confidence between 0 and 1 to indicate how likely that particular fluent can be detected in the current environment. Thus, it is unclear whether the pre-

vious work is able to handle representations with uncertainties. Our interactive learning approach aims to address these uncertainties through interactive question answering with human partners.

6.3 Interactive Learning

6.3.1 Framework of Interactive Learning

Figure 6.4 shows a general framework for interactive learning of action verbs. It aims to support a life-long learning cycle for robots, where the robot can continuously (1) engage in collaboration and communication with humans based on its existing knowledge; (2) acquire new verbs by learning from humans and experiencing the change of the world (i.e., grounded verb semantics as in this work); and (3) learn how to interact (i.e., update interaction policies). The lifelong learning cycle is composed by a sequence of interactive learning episodes (Episode 1, 2...) where each episode consists of either an execution phase or a learning phase or both.

The execution phase starts with a human request for action (e.g., *boil the water*). According to its interaction policy, the robotic agent may choose to ask one or more questions (i.e., Q_i^+) and wait for human answers (i.e., A_i^+), or select a hypothesis from its existing knowledge base to execute the command (i.e., *Execute*). With the human feedback of the execution, the robot can update its interaction policy and existing knowledge.

In the learning phase, the robot can initiate the learning by requesting a demonstration from the human. After the human performs the task, the robotic agent can either choose to update its knowledge if it feels confident, or it can choose to ask the human one or more questions before updating its knowledge.

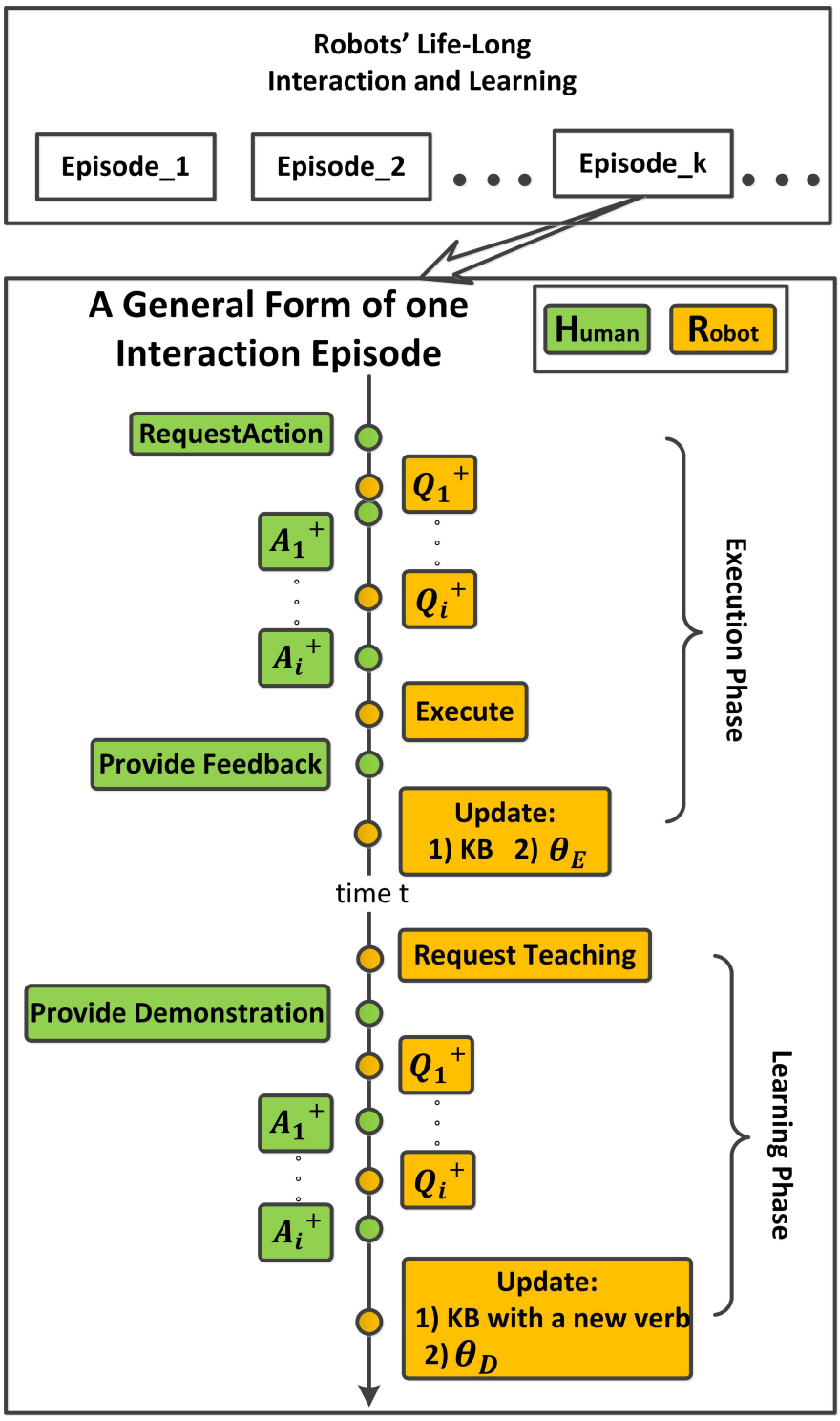


Figure 6.4 A general framework of robot interactive learning. KB stands for knowledge base, θ_E stands for Interaction Strategy for Execution, and θ_D stands for Interaction Strategy for Learning.

Only demonstration[1]	Interactive learning (this work)
Execution Example	**Execution Example**
H1: Boil the water. R1: (*select a hypothesis and execute the action accordingly*) H2: (*provide true/false feedback on whether the execution is* *correct*)	H1: Boil the water. R1: I see a silver object on the table, is that a pot? H2: Yes. R2: Should I leave the pot on the stove after boil? H3: Yes. R3: Ok. (*select a hypothesis to execute*) H4: Thank you.
Learning Example	**Learning Example**
H3: Boil the water. R3: I don't know how to boil, could you show me an example? H4: (*human shows the robot how to achieve the corresponding action through a sequence of primitive actions*) R4: (*indicate the acceptance and induce a hypothesis to rep-* *resent verb semantics*)	H5: Boil the water. R5: I don't know how to boil, could you show me an example? H6: Ok. (*demonstrate the task, at the end put the pot on a heat pad*) R6: So after boiling something, should I put it on a heat pad? H7: No, you don't have to. R7: I see. (*induce goal state hypothesis*)

Table 6.1 Examples to show differences between learning through demonstrations as in the previous works [1] and the proposed learning from interaction.

6.3.2 Examples of Interactive Learning

Table 6.1 illustrates the differences between the previous approach that acquires verb models based solely on demonstrations and our current work that acquires models based on interactive learning. As shown in Table 6.1, under the *demonstration* setting, humans only provide a demonstration of primitive actions and there's no interactive question answering. In the *interactive learning* setting, the robot can proactively choose to ask questions regarding the uncertainties either about the environment (e.g., R1), the goal (e.g., R2), or the demonstrations (e.g., R6). Our hypothesis is that rich interactions based on question answering will allow the robot to learn more reliable models for grounded verb semantics, especially in a noisy environment.

Then the question is how to manage such interaction: when to ask and what questions to ask to

most efficiently acquire reliable models and apply them in execution. Next we describe the application of reinforcement learning to manage interactive question answering for both the execution phase and the learning phase.

6.3.3 Formulation of Interactive Learning

Markov Decision Process (MDP) and its closely related Reinforcement Learning (RL) have been applied to sequential decision-making problems in dynamic domains with uncertainties, e.g., dialogue/interaction management [79, 80, 81], mapping language commands to actions [32], interactive robot learning [82], and interactive information retrieval [83]. In this work, we formulate the choice of when to ask what questions during interaction as a sequential decision-making problem and apply reinforcement learning to acquire an optimal policy to manage interaction.

Specifically, each of the execution and learning phases is governed by one policy (i.e., θ_E and θ_D), which is updated by the reinforcement learning algorithm. The use of RL intends to obtain optimal policies that can lead to the highest long-term reward by balancing the cost of interaction (e.g., the length of interaction and difficulties of questions) and the quality of the acquired models. The reinforcement formulation for both the execution phase and the learning phase are described below.

State For the execution phase, each state $s_e \in S_E$ is a five tuple: $s_e = <l, e, KB, Grd, Goal>$. l is a language command, including a verb and multiple noun phrases extracted by the Stanford parser. For example, the command "*Microwave the ramen*" is represented as $l = microwave(ramen)$. The environment e is a probabilistic representation of the currently perceived physical world, consisting of a set of grounded fluents and the confidence of perceiving each fluent (an example is shown in Figure 6.3). KB stands for the existing knowledge of verb models. Grd accounts for the agent's current belief of object grounding: the probability of each noun in the l being grounded to different

Action Name	Explanation	Question Example	Reward
1. **np_grd_whq**(n)	Ask for the grounding of a np.	"*Which is the cup, can you show me?*"	-6.5[i]
2. **np_grd_ynq**(n, o)	Confirm the grounding of a np.	"*I see a silver object, is that the pot?*"	-1 / -2
3. **env_pred_ynq**(p)	Confirm a predicate in current environment.	"*Is the microwave door open?*"	-1 / -2
4. **goal_pred_ynq**(p)	Confirm whether a predicate p should be in the final environment.	"*Is it true the pot should be on the counter?*"	-1 / -2
5. **select_hypo**(h)	Choose a hypothesis to use as goal and execute.		100 / -2
6. **bulk_np_grd_ynq**(n, o)	Confirm the grounding of multiple nps.	"*I think the pot is the red object and milk is in the white box, am I right?*"	-3 / -6[ii]
7. **pred_change_ynq**(p)	Ask whether a predicate p has been changed by the action demonstration.	"*The pot is on a stand after the action, is that correct?*"	-1 / -2
8. **include_fluent**($\wedge p$)	Include $\wedge p$ into the goal state representation. Update the verb semantic knowledge.		100 / -2

Table 6.2 The action space for reinforcement learning, where n stands for a noun phrase, o a physical object, p a fluent representation of the current state of the world, h a goal hypothesis. Action 1 and 2 are shared by both the execution and learning phases. Action 3, 4, 5 are for the execution phase, and 6, 7, 8 are only used for the learning phase. -1/-2 are typically used for yes/no questions. When the human answers the question with a "yes", the reward is -1, otherwise it's -2.

objects. $Goal$ represents the agent's belief of different goal state hypotheses of the current command. Within one interaction episode, command l and knowledge KB will stay the same, while e, Grd, and $Goal$ may change accordingly due to interactive question answering and robot actions. In the execution phase, Grd and $Goal$ are initialized with existing knowledge of learned verb models. For the learning phase, a state $s_d \in S_D$ is a four tuple: $s_d = <l, e_{start}, e_{end}, Grd>$. e_{start} and e_{end} stands for the environment before the demonstration and after the demonstration.

Action Motivated by previous studies on how humans ask questions while learning new skills [4],

the agent's question set includes two categories: yes/no questions and wh- questions. These questions are designed to address ambiguities in noun phrase grounding, uncertain environment sensing, and goal states. They are domain independent in nature. For example, one of the questions is np_grd_ynq(n, o). It is a yes/no question asking whether the noun phrase n refers to an object o (e.g., "*I see a silver object, is that the pot?*"). Other questions are env_pred_ynq(p) (i.e., whether a fluent p is present in the environment; e.g., "*Is the microwave door open?*") and goal_pred_ynq(p) (i.e., whether a predicate p should be part of the goal; "*Should the pot be on a pot stand?*"). Table 6.2 lists all the actions available in the execution and learning phases. The select_hypo action (i.e., select a goal hypothesis to execute) is only for the execution. Ideally, after asking questions, the agent should be more likely to select a goal hypothesis that best describes the current situation. For the learning phase, the include_fluent($\wedge p$) action forms a goal hypothesis by conjoining a set of fluents ps where each p should have high probability of being part of the goal.

Transition The transition function takes action a in state s, and gives the next state s' according to human feedback. Note that the command l does not change during interaction. But the agent's belief of environment e, object grounding Grd, and goal hypotheses $Goal$ is changed according to the questions and human answers. For example, suppose the agent asks whether noun phrase n refers to the object o, if the human confirms it, the probability of n being grounded to o becomes 1.0, otherwise it will become 0.0.

Reward Finding a good reward function is a hard problem in reinforcement learning. Our current approach has followed the general practice in the spoken dialogue community [84, 85, 86]. The immediate robot questions are assigned small costs to favor shorter and more efficient interaction.

[i] According to the study in [4], the frequency of y/n questions used by humans is about 6.5 times the frequency of open questions (wh question), which motivates our assignment of -6.5 to wh questions.

[ii] bulk_np_grd_ynq asks multiple object grounding all at once. This is harder to answer than asking for a single np. Therefore, its cost is assigned three times of the other yes/no questions.

Furthermore, motivated by how humans ask questions [4], yes/no questions are easier for a human to answer than the open questions (e.g., wh-questions) and thus are given smaller costs. A large positive reward is given at the end of interaction when the task is completed successfully. Detailed reward assignment for different actions are shown in Table 6.2.

Input : e, l (, e_{end});
Feature function ϕ;
Old policy θ (i.e., a weight vector)
Verb Goal States Hypotheses $\mathcal{H}$;
Initialize: state s initialized with e, l (, e_{end});
first action $a \sim P(a|s;\theta)$ with ϵ greedy
while *s is not terminal* **do**
  Take action a, receive reward r;
  $s' = T(s,a)$;
  Choose $a' \sim P(a'|s';\theta)$ with ϵ greedy; $\delta \leftarrow r + \gamma \cdot \theta^T \cdot \phi(s',a') - \theta^T \cdot \phi(s,a)$;
  $\theta \leftarrow \theta + \delta \cdot \eta \cdot \phi(s,a)$;
end
if *s terminates with positive feedback* **then**
  Update $\mathcal{H}$;
end
Output : Updated $\mathcal{H}$ and θ.

Figure 6.5 Policy learning. The execution and learning phases share the same learning process, but with different state s, action a spaces, and feature vectors ϕ. The e_{end} is only available to the learning phase.

Learning The *SARSA* algorithm with linear function approximation is utilized to update policies θ_E and θ_D [87]. Specifically, the objective of training is to learn an optimal value function $Q(s,a)$ (i.e., the expected cumulative reward of taking action a in a state s). This value function is approximated by a linear function $Q(s,a) = \theta^\intercal \cdot \phi(s,a)$, where $\phi(s,a)$ is a feature vector and θ is a weight updated during training. Details of the algorithm is shown in Figure 6.5. During testing, the agent can take an action a that maximizes the Q value at a state s.

Feature Example features used by the two phases are listed in Table 6.3. These features intend to capture different dimensions of information such as specific types of questions, how well

Features shared by both phases
If a is a np_grd_whq(n).
The entropy of candidate groundings of n.
If n has more than 4 grounding candidates.
If a is a np_grd_ynq(n, o).
The probability of n grounded to o.
Additional Features specific for the Execution phase
If a is a select_hypo(h) action.
The probability of hypo h not satisfied in current environment.
Similarity between the ns used by command l and the commands from previous experiences.
Additional Features specific for the Learning phase
If a is a pred_change_ynq(p).
The probability of p been changed by demo.

Table 6.3 Example features used by the two phases. a stands for action. Other notations are the same as used in Table 6.2. The"If" features are binary, and the other features are real-valued.

noun phrases are grounded to the environment, uncertainties of the environment, and consistencies between a hypothesis and the current environment.

6.4 Evaluation

6.4.1 Experiment Setup

Dataset. To evaluate our approach, we utilized the benchmark made available by [3]. Individual language commands and corresponding action sequences are extracted similarly as [1]. This dataset includes common tasks in the kitchen and living room domains, where each data instance comes with a language command (e.g., "*boil the water*", "*throw the beer into the trashcan*") and the corresponding sequence of primitive actions. In total, there are 979 instances, including 75 different verbs and 215 different noun phrases. The length of primitive action sequences range from 1 to 51 with an average of 4.82 (+/-4.8). We divided the dataset into three groups: (1) 200 data instances were used by reinforcement learning to acquire optimal interaction policies; (2) 600 data instances

were used by different approaches (i.e., previous approaches and our interactive learning approach) to acquire grounded verb semantics models; and (3) 179 data instances were used as testing data to evaluate the learned verb models. The performance on applying the learned models to execute actions for the testing data is reported.

To learn interaction policies, a simulated human model is created from the dataset [84] to continuously interact with the robot learner[iii]. This simulated user can answer the robot's different types of questions and make decisions on whether the robot's execution is correct. During policy learning, one data instance can be used multiple times. At each time, the interaction sequence is different due to *exploitation and exploration* in RL in selecting the next action. The RL discount factor γ is set to 0.99, the ϵ in ϵ- greedy is 0.1, and the learning rate is 0.01.

Noisy Environment Representation. The original data provided by [3] is based on the assumption that environment sensing is perfect and deterministic. To enable incomplete and noisy environment representation, for each fluent (e.g., $grasping(Cup_3)$, $near(robot_1, Cup_3)$) in the original data, we independently sampled a confidence value to simulate the likelihood that a particular fluent can be detected correctly from the environment. We applied the following four different variations in sampling the confidence values, which correspond to different levels of sensor reliability.

(1) PerfectEnv represents the most reliable sensor. If a fluent is true in the original data, its sampled confidence is 1, and 0 otherwise.

(2) NormStd3 represents a relatively reliable sensor. For each fluent in the original environment, a confidence is sampled according to a normal distribution $\mathcal{N}(1, 0.3^2)$ with an interval [0,1]. This distribution has a large probability of sampling a number larger than 0.5, meaning the corresponding

[iii] In future work, interacting with real humans can be conducted through Amazon Mechanical Turk. And the policies acquired with a simulated user in this work will be used as initial policies.

fluent is still more likely to be true.

(3) NormStd5 represents a less reliable sensor. The sampling distribution is $\mathcal{N}(1, 0.5^2)$, which has a larger probability of generating a number smaller than 0.5 compared to *NormStd3*.

(4) UniEnv represents an unreliable sensor. Each number is sampled with a uniform distribution between 0 and 1. This means the sensor works randomly. A fluent has a equal change to be true or false no matter what the true environment is.

Evaluation Metrics. We used the same evaluation metrics as in the previous works [1, 3] to evaluate the performance of applying the learned models to testing instances on action planning.

- ***IED***: Instruction Editing Distance. This is a number between 0 and 1 measuring the similarity between the predicted action sequence and the ground-truth action sequence. ***IED*** equals 1 if the two sequences are exactly the same.

- ***SJI***: State Jaccard Index. This is a number between 0 and 1 measuring the similarity between the predicted and the ground-truth state changes. ***SJI*** equals 1 if action planning leads to exactly the same state change as in the ground-truth.

Configurations. To understand the role of interactive learning in model acquisition and action planning, we first compared the interactive learning approach with the previous leading approach (presented as *She16*). To further evaluate the interaction policies acquired by reinforcement learning, we also compared the learned policy (i.e., ***RLPolicy***) with the following two baseline policies:

- ***RandomPolicy***: randomly selects questions to ask during interaction.

- ***ManualPolicy***: continuously asks for yes/no confirmations (i.e., object grounding questions ($GroundQ$), environment questions ($EnvQ$), goal prediction questions ($GoalQ$)) until there's no more questions before making a decision on model acquisition or action execution.

	She16		*RL policy*		% improvement	
	IED	*SJI*	*IED*	*SJI*	*IED*	*SJI*
PerfectEnv	0.430	0.426	0.453	0.468	5.3%*	9.9%*
NormStd3	0.284	0.273	0.420	0.431	47.9%*	57.9%*
NormStd5	0.172	0.168	0.392	0.411	127.9%*	144.6%*
UniEnv	0.168	0.163	0.332	0.347	97.6%*	112.9%*

Table 6.4 Performance comparison between *She16* and our interactive learning based on environment representations with different levels of noise. All the improvements (marked *) are statistically significant ($p < 0.01$).

For the *ManualPolicy*, we manually defined the priority of questions to be asked. Specifically, in accordance with the pipeline of the system process, the three categories questions are asked in following order: $GroundQ$ -> $EnvQ$ -> $GoalQ$. For example, the $GroundQ$ will be asked first. When the agent is certain about the groundings of all the noun phrases (e.g., based on the human answers, the agent will know if the object asked in the question is the true grounding or not), it will begin to ask environment related questions, and then goal state questions. Within each category, there are multiple candidate questions that can be asked, and the agent will first choose to ask the question that has the highest probability (e.g., For $GroundQ$s, this probability is calculated based on the frequency of an object being referred with the noun phrase in previous experiences. For $EnvQ$s, this is the probability of a specific predicate sensed to be true in the current environment).

6.4.2 Results

6.4.2.1 The Effect of Interactive Learning

Table 6.4 shows the performance comparison on the testing data between the previous approach *She16* and our interactive learning approach based on environment representations with different levels of noise. The verb models acquired by interactive learning perform better consistently across

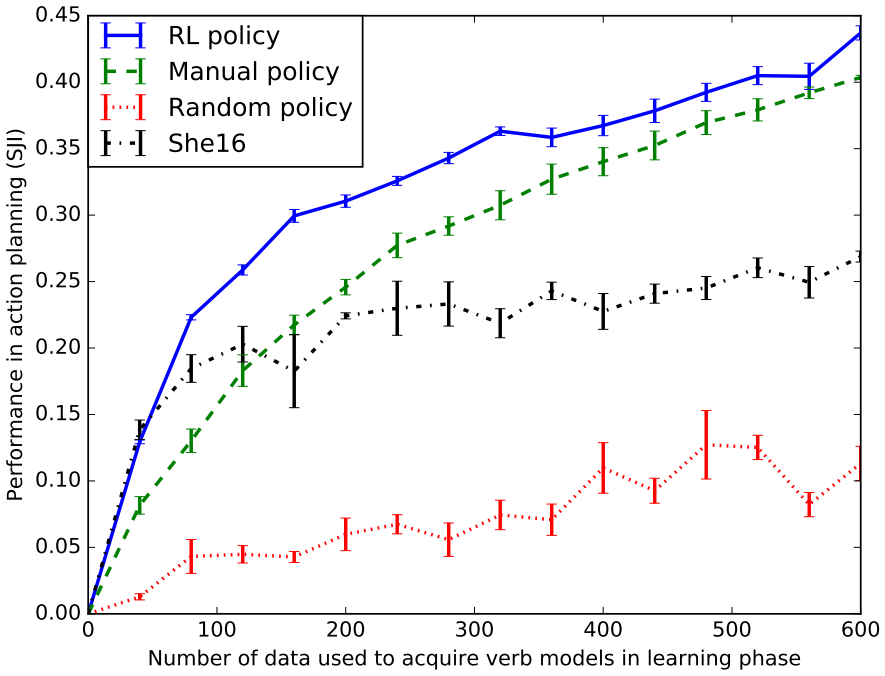


Figure 6.6 Performance (SJI) comparison by applying models acquired based on different interaction policies to the testing data.

all four environment conditions. When the environment becomes noisy (i.e., *NormStd3*, *NormStd5*, and *UniEnv*), the performance of *She16* that only relies on demonstrations decreases significantly. While the interactive learning improves the performance under the perfect environment condition, its effect in noisy environment is more remarkable. It leads to a significant performance gain between 48% and 145%. These results validate our hypothesis that interactive question answering can help to alleviate the problem of uncertainties in environment representation and goal prediction.

Figure 6.6 shows the performance of the various learned models on the testing data, based on a varying number of training instances and different interaction policies. The interactive learning guided by the policy acquired from RL outperforms the previous approach *She16*. The RL policy slightly outperforms interactive learning using manually defined policy (i.e., *ManualPolicy*). However, as shown in the next section, the *ManualPolicy* results in much longer interaction (i.e., more

	Average number of questions							Performance	
	Learning Phase			Execution Phase					
	$GroundQ$	$EnvQ$	*TotalQ*	$GroundQ$	$EnvQ$	$GoalQ$	*TotalQ*	IED	SJI
RL Policy	2.130* +/-0.231	2.615* +/-0.317	4.746* +/-0.307	0.383* +/-0.137	0.650* +/-0.366	2.626 +/-0.331	3.665* +/-0.469	0.420 +/-0.015	0.430* +/-0.018
Manual Policy	2.495 +/-0.025	5.338 +/-0.008	7.833 +/-0.025	1.236 +/-0.002	3.202 +/-0.012	2.353 +/-0.023	6.792 +/-0.025	0.406 +/-0.002	0.404 +/-0.004
Random Policy	0.545 +/-0.016	0.368 +/-0.033	0.913 +/-0.040	0.678 +/-0.055	0.081 +/-0.030	0.151 +/-0.024	0.909 +/-0.018	0.114 +/-0.025	0.113 +/-0.029

Table 6.5 Comparison between different policies including the average number (and standard deviation) of different types of questions asked during the execution phase and the learning phase respectively, and the performance on action planning for the testing data. The results are based on the noisy environment sampled by *NormStd3*. * indicates statistically significant difference ($p < 0.05$) comparing *RLPolicy* with *ManualPolicy*.

questions) than the RL acquired policy.

6.4.2.2 Comparison of Interaction Policies

Table 6.5 compares the performance of different interaction policies. It shows the average number of questions asked under different policies. It is not surprising the *RandomPolicy* has the worst performance. For the *ManualPolicy*, its performance is similar to the *RLPolicy*. However, the average interaction length of *ManualPolicy* is 6.792, which is much longer than the *RLPolicy* (which is 3.127). These results further demonstrate that the policy learned from RL enables efficient interactions and the acquisition of more reliable verb models.

6.4.3 Transfer the Interaction Policies to a Physical Robot

To demonstrate the generality of interaction policy, we transferred the policies acquired from virtual world into a physical robot (i.e., the Baxter robot). A demo [ii] is created to explicitly illustrate the

[ii] https://youtu.be/wOiFRCLTX8M

learned policy. The demo is to simulate a kitchen environment, where food (e.g., juice bottles, a cereal box) and cookwares (e.g., kettle, portable burner, and a cutting board) are placed on a table top. The Baxter robot stands behind the table, and can reach every object on the table. In the demo, the human issues a command "H: *place the pot on the burner.*". The current robot doesn't understand the verb "*place*", so it asks the human to demonstrate how to perform this action. And the human quickly show this action. From this demonstration, the robot has initial understanding about how the environment has been changed, as well as the connections between noun phrases and objects on the table. But from the intermediate information, we can see that the robot's understanding are ambiguous, meaning the robot is not "confident" about its understanding. Directed by the interaction policy, the robot automatically chooses to clarify these ambiguities before making further decisions of inducing action knowledge, for example, pointing and asking human questions (e.g., "R: *What you say burner, do you mean this?*"). In the end, Baxter uses the just acquired knowledge to re-do this action and asks for human feedback. Only if the human accepts the reproduced action, Baxter can finally accept that the just acquired action is correct and can be safely stored in its knowledge for future use.

A screen shot of the demo [ii] is shown in Figure 6.7. The interface includes an image section (i.e., displaying the perceived environment from the view point of the human) and a set of information sections (i.e., showing the robot's intermediate understanding about the interaction). These intermediate information includes the language understanding, the environment sensing, initial understanding of the human demonstration, and the robot's decision making process. Specifically, the information sections explicitly illustrate following content:

- Dialogue History: records the sentences from both the human and the robot itself.
- Environment: displays the robot's understanding of the current environment, in terms of

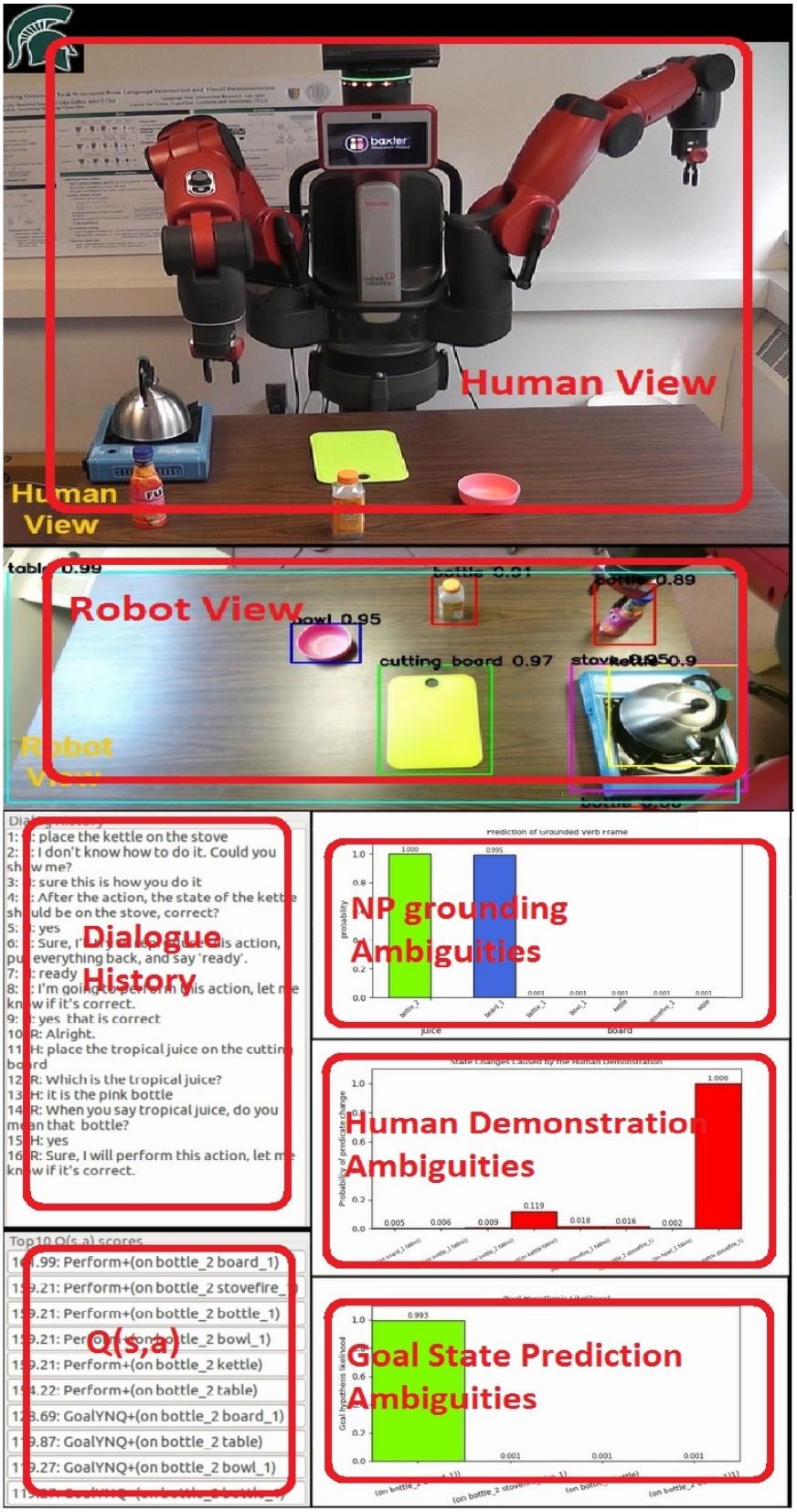


Figure 6.7 A screenshot of the Baxter learning demo.

detected state fluents and the probability of detecting each fluent.

- Top10 Q(s,a): shows the top 10 system actions that is available to the robot at each dialogue turn. The number before each action stands for Q(s,a) (i.e., the Q-value of executing action a in current state s), which is calculated based on parameters updated through reinforcement learning. In this demo, we assume the interaction policy is fixed (i.e., parameters of calculating Q(s,a) have already been acquired in previous interactions), and, for now, each time the robot always greedily choose the top1 action to act.

- Language Understanding: includes the verb and noun phrases extracted by Stanford Parser [88], as well as the probability of each noun phrases refers to different objects in the scene. These np-object grounding probabilities are further illustrated through a bar chart. Throughout the interaction, changes of the robot's understanding can be directly reflected through the bar chart.

- Demonstration Understanding: is reflected by the understanding of state changes caused by the human demonstration. Specifically, the robot compares the environments before and after the demonstration to capture the fluents that have been changed, as well as the probability of each fluent change. Because the environment sensing is noisy, these extracted state changes may not always be true. Through asking questions and getting feedback from the human, the robot can modify its initial understanding about these state changes. All these informations are reflected in the State Changes section.

- Most Recently Acquired Verb Semantic: displays the new action knowledge added to the robot's action knowledge base. This information is updated only when the human accepts the robot's reproduction of a acquired knowledge.

To transfer the knowledge (i.e., interaction policy, np-object grounding information, and action knowledge) acquired through virtual data onto the physical robot, following issues should be taken into consideration:

1. Environment sensing: The set of predicates used to represent the physical environment should be included in the predicate set used the virtual data. The main reason is that: the action knowledge (i.e., verb goal states) are represented by conjunctions of state predicates. Any verb goal state with a predicate that's not in the physical environment representation is not applicable to the physical environment.

2. NP-Object grounding: Currently, the robot's knowledge of NP-Object are bounded with "object instances" meaning that the robot remembers the probabilistic mappings between noun phrases and object IDs. In the physical environment, the same IDs should be used to identify physical objects. If an object o in the physical environment is identified by an ID that's not included in the robot knowledge, it will be not be verbally recognized in terms of what noun phrases are used to refer to this object o.

6.5 Conclusion

In previous chapters, we have investigated to represent verb semantics with goal state based representations, either single goal state or a hypothesis space of goals. The previous procedure of acquiring verb goals assumes perfect and deterministic environment sensing. However, robots live in a noisy environment. Due to the limitations in their external sensors, their representations of the shared environment can be error prone and full of uncertainties. Learning action models from the noisy and incomplete observation of the world is extremely challenging. To address this problem, this chapter presents an interactive learning approach which aims to handle uncertainties of the en-

vironment as well as incompleteness and conflicts in state representation by asking human partners intelligent questions. The interaction strategies are learned through reinforcement learning. Our empirical results have shown a significant improvement in model acquisition and action prediction. When applying the learned models in new situations, the models acquired through interactive learning lead to over 140% performance gain in noisy environment.

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Recent years have seen an increasing interest in AI personal assistant working around people. These AI agents may be softwares embedded in portable devices (e.g., Alexia, Siri, or Cortana), or some humanoid physical robots (e.g., NAO, Baxter, and ASIMO). Most of these intelligent agents, to some extent, have the capability of establishing general purpose dialogue with humans, understanding certain language commands, and perform actions accordingly. However, most of these intelligent agents lack the capability of extending their knowledge. In another word, they are not good at learning more knowledge when working with real users, on top of their "built in" knowledge. The capability of understanding and performing new commands is very important, because end-users may always use language that's out of the robots' vocabulary. End-users may also want to have personalized functions (e.g., learn the layout of the end-user's house, cook based on the user's taste, clean the house based on how the user organizes different rooms). These personalized actions cannot be designed in the factory, and have to be learned from each individual user. Towards this goal, this thesis investigates the problem of learning to understand and perform new actions through dialogue/interaction with a human user, where we have made the following contributions:

- In Chapter 3, a goal state based verb semantics is proposed to bridge the gap of symbolic human language and continuous robotic movements. Specifically, in accordance with result verbs, verb semantics is represented by a conjunction of environment fluents, capturing the

desired goal state of executing this verb action. Moreover, based on the verb goal representation, a three-tier action knowledge base is proposed. In the lowest level of the knowledge base are the primitive actions the robot can perform, in terms of basic movement trajectory definitions. In the highest level are verb goal state information. In the middle is a set of action schemas for planning. Each operation is a structure capturing the preconditions and result of performing a low level primitive action. A symbolic planner (e.g., PDDL planner) is used to calculate a sequence of primitive actions to achieve the highest level goal from the current environment. As shown in an empirical evaluation, the acquired verb goal state can be successfully applied to generate action sequences in new environment configurations.

- Chapter 4 discusses an approach of acquiring verb goal state through situated dialogue with a human partner. Specifically, when the robot doesn't understand a verb, it will ask the human to teach it. The human will provide step-by-step instructions. If the instruction involves new unknown verbs, the robot will initiate a sub-dialogue to learn the semantics of the unknown verb. By comparing the environment before and after the teaching, the robot can extract the changed states to form new action knowledge. A dialogue system is proposed and implemented on a robot arm (i.e., SCHUNK arm). An empirical evaluation demonstrates the system learning capability by learning new verbs in a simple blocks-world and applying the acquired knowledge in novel situations. Furthermore, when human teaches the robot through step-by-step instructions, the teaching process is longer compared with the one-shot instructions, but it leads to better learning performance.

- In Chapter 5, we propose a verb goal state hypothesis space to make the verb representation more flexible. A regression based hypothesis selector is used to choose a proper goal of a verb according to the execution environment. In particularly, to handle the different outcomes

of a verb, we use a specific-to-general hierarchical space of goal state hypothesis to represent the semantics of a verb. Each hypothesis in the space is one possible outcome of this action verb. And a hypothesis with a more detailed description of the outcome (i.e., more specific) locates at the lower levels of the hierarchy. During executing the action, a regression based selector is used to calculate the score of appropriateness of executing a particular hypothesis in the current environment, and the hypothesis with the highest score is used for the robot to perform. Furthermore, the incremental nature of learning hypothesis space better supports life-long learning. The evaluation on a dataset collected from a simulated environment has shown that the verb goal hypothesis space, combined with the learned hypothesis selector, outperforms the previous leading approach [18] in terms of execution prediction accuracy.

- Chapter 6 focuses on the problem of learning under uncertain sensing. Specifically, approaches discussed in previous sections are built on the assumption of deterministic and perfect sensing setup. However, in real-world applications, the robot's understanding of the environment is always noisy and ambiguous. On the other hand, the rich interaction between the teacher and students has not been well explored. To address limitations, in Chapter 6, we propose a new interactive learning approach allowing the robot to be able to actively engage in the learning process. In particular, the new intelligent agent can automatically decide and select questions to ask, guided by the interaction policies acquired through reinforcement learning. As shown by the evaluation results, the proposed interactive learning acquires more reliable action models which lead to higher action sequence prediction accuracy during testing, especially when the environment is noisy and full of uncertainties.

7.2 Future Directions

This dissertation explores several approaches for robots to learn new verbs and actions through language communication with humans. To extend these approaches to a variety of real-world applications, future work should address the following limitations:

- **Representing Environment with a Fixed Set of Fluents:** Currently, the system understanding of the environment or the dynamic of the environment is still based on the close-word assumption. And the acquired action goal state knowledge is based on a set of existing fluents. If an action leads to an environment change that's not captured by the fluents: firstly, this change will not be identified from the robot's point of view; and second, the action model cannot be learned correctly by the system. Despite the need of extending environment representation, automatically identifying what should be perceived from an environment is still difficult. One possible solution is to take advantage of the rich interaction with humans. For example, if no environment changes can be detected based on the human demonstration, it may indicate that certain dynamics of the environment is not covered by the current set of fluents. In such cases, the agent can choose to ask the human if something is missing in its understanding (e.g., "I don't see any changes of the environment, did I miss something?"). And with the human feedback, information about the missing fluents can be gathered.

- **Reward Function in Reinforcement Learning:** In Section 6, the definition of reward functions in reinforcement learning follows the literature of spoken dialogue system, where a large reward is set for the final successful prediction and a small penalty is set to each intermediate interaction step to prefer shorter interaction. Although this reward function setup is widely used by researchers in the dialogue community, how to design reward functions and the relation between reward functions and optimal policy remain challenging problems.

- **Feature Engineering in Function Approximation:** The next limitation is about feature engineering. Specifically, because of the large continuous space and the difficulty of deriving a close form solution, the Q-value functions in reinforcement learning are always approximated by certain functions. In our investigation, we utilize linear functions to approximate the Q(s,a), where each linear function is a weighted summation of a set of features. Currently these features are pre-defined intended to capture useful informations from different aspects of the interaction. One possible future direction is to use deep neural network to alleviate the problem of feature engineering.

- **Mechanical Limitation of Physical Robots:** Another limitation of transferring the knowledge acquired in virtual environment to the physical world is how to implement the robot's primitive movements. In a virtual world, the environment dynamics and robot movements are much easier to control comparing with the physical world. In the virtual world, we can click a button to realize the function of attaching an object to the robot's hand meaning that the robot is grasping the object. However, in the physical world, even a "simple" grasping action may involve different issues to robustly perform. Firstly, finding a trajectory that can make the rigid arm avoid any obstacles can be hard. Secondly, for an unknown object, where should the robot gripper grasp is another research problem. Thirdly, even for the same object (e.g., a bottle), the robot may need to grasp different parts of the object based on the goal (e.g., If it just wants to pick up a bottle, the robot can grasp the top part. But if the robot wants to pour water from the bottle, it should grasp the middle part). The limitation of robust low level movements tends to be ignored by people working on higher level knowledge, but it is another major challenge to use robots in real-world applications. An interactive approach that can reduce the number of demonstrations required to acquire low-level movements can

lead to a great impact.

As cognitive robots start to enter our daily lives, data-driven approaches to learning may not be widely applicable in new situations. Human partners who work side-by-side with these cognitive robots are great resources that the robots can directly learn from. Despite the efforts we made in this dissertation, many important and interesting problems still remain. We believe that future research on this topic is of great value to the AI related communities.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] L. She and J. Y. Chai, "Incremental acquisition of verb hypothesis space towards physical world interaction," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[2] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," in *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, (San Francisco, CA, USA), pp. 608–620, Morgan Kaufmann Publishers Inc., 1971.

[3] D. K. Misra, K. Tao, P. Liang, and A. Saxena, "Environment-driven lexicon induction for high-level instructions," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Beijing, China), pp. 992–1002, Association for Computational Linguistics, July 2015.

[4] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, (New York, NY, USA), pp. 17–24, ACM, 2012.

[5] M. R. Hovav and B. Levin, "Reflections on Manner / Result Complementarity," *Lexical Semantics, Syntax, and Event Structure*, pp. 21–38, 2010.

[6] M. R. Hovav and B. Levin, "Reflections on manner/result complementarity," *Lecture notes*, 2008.

[7] C. Kennedy and L. McNally, "Scale structure, degree modification, and the semantics of gradable predicates," *Language*, pp. 345–381, 2005.

[8] K. K. Schuler, "Verbnet: A broad-coverage, comprehensive verb lexicon," 2005.

[9] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," *Cognitive Psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[10] P. Gorniak and D. Roy, "Situated language understanding as filtering perceived affordances," *Cognitive Science*, vol. Volume31(2), pp. 197–231, 2007.

[11] S. Tellex, P. Thaker, J. Joseph, and N. Roy, "Learning perceptually grounded word meanings from unaligned parallel data," *Machine Learning*, vol. Volume94, no. 2, pp. 151–167, 2014.

[12] J. Kim and R. J. Mooney, "Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision," in *Proceedings of the EMNLP-CoNLL '12*, pp. 433–444,

2012.

[13] C. Matuszek, N. Fitzgerald, L. Zettlemoyer, L. Bo, and D. Fox, "A joint model of language and perception for grounded attribute learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)* (J. Langford and J. Pineau, eds.), (New York, NY, USA), pp. 1671–1678, ACM, 2012.

[14] C. Liu, L. She, R. Fang, and J. Y. Chai, "Probabilistic labeling for efficient referential grounding based on collaborative discourse," in *Proceedings of the ACL'14 (Volume 2: Short Papers)*, pp. 13–18, 2014.

[15] C. Liu and J. Y. Chai, "Learning to mediate perceptual differences in situated human-robot dialogue," in *Proceedings of the AAAI'15*, pp. 2288–2294, 2015.

[16] J. Thomason, S. Zhang, R. Mooney, and P. Stone, "Learning to interpret natural language commands through human-robot dialog," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1923–1929, July 2015.

[17] J. Thomason, J. Sinapov, M. Svetlik, P. Stone, and R. J. Mooney, "Learning multi-modal grounded linguistic semantics by playing "i spy"," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-16)*, (New York City), pp. 3477–3483, 2016.

[18] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me dave: Context-sensitive grounding of natural language to manipulation instructions," *Proceedings of Robotics: Science and Systems (RSS), Berkeley, USA*, 2014.

[19] C. Matuszek, E. Herbst, L. S. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system.," in *ISER* (J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, eds.), vol. 88 of *Springer Tracts in Advanced Robotics*, pp. 403–415, Springer, 2012.

[20] Y. Artzi and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," *Transactions of the Association for Computational Linguistics*, vol. Volume1, no. 1, pp. 49–62, 2013.

[21] L. She, S. Yang, Y. Cheng, Y. Jia, J. Y. Chai, and N. Xi, "Back to the blocks world: Learning new actions through situated human-robot dialogue," in *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pp. 89–97, 2014.

[22] D. L. Chen and R. J. Mooney, "Learning to interpret natural language navigation instructions from observations.," in *Proceedings of the AAAI'11*, pp. 859–865, 2011.

[23] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: A survey," *IEEE Transactions on Autonomous Mental Development*, vol. 1, no. 1, pp. 12–34, 2009.

[24] A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller, "Interactive hierarchical task learning from a single demonstration," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pp. 205–212, ACM, 2015.

[25] N. Nejati, P. Langley, and T. Konik, "Learning hierarchical task networks by observation," in *Proceedings of the 23rd international conference on Machine learning*, pp. 665–672, ACM, 2006.

[26] C. Liu, S. Yang, S. Saba-Sadiya, N. Shukla, Y. He, S.-c. Zhu, and J. Chai, "Jointly learning grounded task structures from language instruction and visual demonstration," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 1482–1492, Association for Computational Linguistics, November 2016.

[27] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers.," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, 2008.

[28] J. M. Siskind, "Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic," *J. Artif. Int. Res.*, vol. 15, pp. 31–90, Feb. 1999.

[29] X. Wang, "Learning by observation and practice: An incremental approach for planning operator acquisition," in *Proceedings of the ICML'95*, pp. 549–557, 1995.

[30] Y. Gil, "Learning by experimentation: incremental refinement of incomplete planning domains," in *Prococeedings of the ICML'94*, pp. 87–95, 1994.

[31] Q. Yang, K. Wu, and Y. Jiang, "Learning action models from plan examples using weighted max-sat," *Artificial Intelligence*, vol. Volume171, no. 2–3, pp. 107 – 143, 2007.

[32] S. R. K. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay, "Reinforcement learning for mapping instructions to actions," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, (Stroudsburg, PA, USA), pp. 82–90, Association for Computational Linguistics, 2009.

[33] S. Branavan, N. Kushman, T. Lei, and R. Barzilay, "Learning high-level planning from text," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Jeju Island, Korea), pp. 126–135, Association for Computational Linguistics, July 2012.

[34] S. R. K. Branavan, L. S. Zettlemoyer, and R. Barzilay, "Reading between the lines: Learning to map high-level instructions to commands," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, (Stroudsburg, PA, USA), pp. 1268–1277, Association for Computational Linguistics, 2010.

[35] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*,

HRI '10, (Piscataway, NJ, USA), pp. 259–266, IEEE Press, 2010.

[36] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation.," in *Proceedings of the AAAI'12*, pp. 1507–1514, 2011.

[37] J. MacGlashan, M. Babes-Vroman, M. desJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang, "Grounding english commands to reward functions.," in *Robotics: Science and Systems*, 2015.

[38] D. L. Chen, J. Kim, and R. J. Mooney, "Training a multilingual sportscaster: Using perceptual context to learn language," *J. Artif. Int. Res.*, vol. 37, pp. 397–436, Jan. 2010.

[39] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[40] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," vol. 67, pp. 469–483, January 2009.

[41] R. Stalnaker, "Common ground," *Linguistics and Philosophy*, vol. 25, pp. 701–721, 2002.

[42] H. H. Clark and S. E. Brennan, "Grounding in communication," in *Perspectives on socially shared cognition* (L. B. Resnick, R. M. Levine, and S. D. Teasley, eds.), pp. 127–149, 1991.

[43] H. H. Clark, *Using language*. Cambridge, UK: Cambridge University Press, 1996.

[44] H. H. Clark and E. F. Schaefer, "Contributing to discourse," in *Cognitive Science*, no. 13, pp. 259–294, 1989.

[45] H. H. Clark and D. Wilkes-Gibbs, "Referring as a collaborative process," in *Cognition*, no. 22, pp. 1–39, 1986.

[46] D. Gergle, R. E. Kraut, and S. R. Fussell, "Using visual information for grounding and awareness in collaborative tasks," *Human Computer Interaction*, vol. 28, pp. 1–39, 2013.

[47] D. Traum, *A Computational Theory of Grounding in Natural Language Conversation*. PhD thesis, University of Rochester, 1994.

[48] S. Kiesler, "Fostering common ground in human-robot interaction," in *Proceedings of the IEEE International Workshop on Robots and Human Interactive Communication (ROMAN)*, pp. 729–734, 2005.

[49] A. Powers, A. Kramer, S. Lim, J. Kuo, S.-L. Lee, and S. Kiesler, "Common ground in dialogue with a gendered humanoid robot," in *Proceedings of ICRA*, 2005.

[50] K. Stubbs, P. Hinds, and D. Wettergreen, "Autonomy and common ground in human-robot

interaction," in *IEEE Intelligent Systems*, pp. 42–50, 2007.

[51] K. Stubbs, D. Wettergreen, and I. Nourbakhsh, "Using a robot proxy to create commmon ground in exploration tasks," in *Proceedings of the HRI08*, pp. 375–382, 2008.

[52] J. Y. Chai, R. Fang, C. Liu, and L. She, "Collaborative language grounding toward situated human-robot dialogue.," *AI Magazine*, vol. 37, no. 4, 2016.

[53] J. Peltason, H. Rieser, S. Wachsmuth, and B. Wrede, "On grounding natural kind terms in human-robot communication," *Künstliche Intelligenz*, vol. 27, pp. 107–118, 2013.

[54] H. Buschmeier and S. Kopp, "Co-constructing grounded symbols - feedback and incremental adaptation in human-agent dialogue," *Künstliche Intelligenz*, vol. 27, pp. 137–143, 2013.

[55] J. F. Allen, N. Chambers, G. Ferguson, L. Galescu, H. Jung, M. D. Swift, and W. Taysom, "Plow: A collaborative task learning agent.," in *AAAI*, pp. 1514–1519, AAAI Press, 2007.

[56] R. Cantrell, P. W. Schermerhorn, and M. Scheutz, "Learning actions from human-robot dialogues.," in *RO-MAN* (H. I. Christensen, ed.), pp. 125–130, IEEE, 2011.

[57] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz, "Tell me when and why to do it! run-time planner model updates via natural language instruction," in *Proceedings of the HRI'12*, pp. 471–478, 2012.

[58] S. Mohan, J. Kirk, and J. Laird, "A computational model for situated task learning with interactive instruction," in *Proceedings of ICCM 2013 - 12th International Conference on Cognitive Modeling*, 2013.

[59] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language.," *Robotics and Autonomous Systems*, vol. 38, no. 3-4, pp. 171–181, 2002.

[60] J. D. Bransford, A. L. Brown, and R. R. Cocking, *How People Learn: Brain, Mind, Experience, and School: Expanded Edition*. Washington, DC: National Academy Press., 2000.

[61] M. Cakmak, C. Chao, and A. L. Thomaz, "Designing interactions for robot active learners.," *IEEE T. Autonomous Mental Development*, vol. 2, no. 2, pp. 108–118, 2010.

[62] H. I. Christensen, G. M. Kruijff, and J. Wyatt, eds., *Cognitive Systems*. Springer, 2010.

[63] J. Y. Chai, L. She, R. Fang, S. Ottarson, C. Littley, C. Liu, and K. Hanson, "Collaborative effort towards common ground in situated human robot dialogue," in *Proceedings of 9th ACM/IEEE International Conference on Human-Robot Interaction*, (Bielefeld, Germany), 2014.

[64] C. Liu, R. Fang, L. She, and J. Chai, "Modeling collaborative referring for situated referential grounding," in *Proceedings of the SIGDIAL 2013 Conference*, (Metz, France), pp. 78–86, 2013.

[65] C. Liu, R. Fang, and J. Chai, "Towards mediating shared perceptual basis in situated dialogue," in *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, (Seoul, South Korea), pp. 140–149, 2012.

[66] S. J. Russell, P. Norvig, J. F. Candy, J. M. Malik, and D. D. Edwards, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[67] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[68] S. Edelkamp and P. Kissmann, "Gamer: Bridging planning and general game playing with symbolic search," *IPC6 booklet, ICAPS*, 2008.

[69] J. Rintanen, "Planning as satisfiability: Heuristics," *Artificial Intelligence*, vol. Volume193, pp. 45–86, 2012.

[70] S. Richter and M. Westphal, "The lama planner: Guiding cost-based anytime planning with landmarks," *J. Artif. Intell. Res. (JAIR)*, vol. 39, pp. 127–177, 2010.

[71] A. L. Thomaz and M. Cakmak, "Learning about objects with human teachers," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, HRI '09, (New York, NY, USA), pp. 15–22, ACM, 2009.

[72] M. Steedman and J. Baldridge, "Combinatory categorial grammar," *Non-Transformational Syntax Oxford: Blackwell*, pp. 181–224, 2011.

[73] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED framework: Object Recognition and Pose Estimation for Manipulation," 2011.

[74] L. She, Y. Cheng, J. Y. Chai, Y. Jia, S. Yang, and N. Xi, "Teaching robots new actions through natural language instructions.," in *RO-MAN*, pp. 868–873, 2014.

[75] H. Zhang, Y. Jia, and N. Xi, "Sensor-based redundancy resolution for a nonholonomic mobile manipulator," in *IROS*, pp. 5327–5332, 2012.

[76] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. Volume29, pp. 309–352, 2007.

[77] K. Mourão, L. S. Zettlemoyer, R. P. A. Petrick, and M. Steedman, "Learning strips operators from noisy and incomplete observations.," in *Proceedings of the UAI'12*, pp. 614–623, 2012.

[78] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. Volume12, pp. 2825–2830, 2011.

[79] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue management with reinforcement learning: Experiments with the njfun system," *Journal of Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002.

[80] T. Paek and R. Pieraccini, "Automating spoken dialogue management design using machine learning: An industry perspective," *Speech Communication*, vol. 50, pp. 716–729, Aug. 2008.

[81] J. D. Williams and G. Zweig, "End-to-end lstm-based dialog control optimized with supervised and reinforcement learning," *arXiv preprint arXiv:1606.01269*, 2016.

[82] W. B. Knox and P. Stone, "Understanding human teaching modalities in reinforcement learning environments: A preliminary report," in *IJCAI 2011 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, July 2011.

[83] J. Li, A. H. Miller, S. Chopra, M. Ranzato, and J. Weston, "Learning through Dialogue Interactions by Asking Questions," *ArXiv e-prints*, Dec. 2016.

[84] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies," *Knowl. Eng. Rev.*, vol. 21, pp. 97–126, June 2006.

[85] R. Fang, M. Doering, and J. Y. Chai, "Collaborative models for referring expression generation in situated dialogue," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, AAAI'14, pp. 1544–1550, AAAI Press, 2014.

[86] P.-H. Su, M. Gasic, N. Mrkšić, L. M. Rojas Barahona, S. Ultes, D. Vandyke, T.-H. Wen, and S. Young, "On-line active reward learning for policy optimisation in spoken dialogue systems," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 2431–2441, Association for Computational Linguistics, August 2016.

[87] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[88] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, (Stroudsburg, PA, USA), pp. 423–430, Association for Computational Linguistics, 2003.