Developing High-level Cognitive Functions for Service Robots

Xiaoping Chen, Jianmin Ji, Jiehui Jiang, Guoqiang Jin, Feng Wang, and Jiongkun Xie
Multi-Agent Systems Lab, School of Computer Science and Technology
University of Science and Technology of China
230026, Hefei, P.R. China
xpchen@ustc.edu.cn, {jizheng,jhjiang,abxeeled,fenggew,devilxjk}@mail.ustc.edu.cn

ABSTRACT

The primary target of this work is human-robot collaboration, especially for service robots in complicated application scenarios. Three assumptions and four requirements are identified. State-of-the-art, general-purpose Natural Language Processing (NLP), Commonsense Reasoning (in particular, ASP), and Robotics techniques are integrated in a layered architecture. The architecture and mechanisms have been implemented on a service robot, Ke Jia. Instead of command languages, small limited segments of natural languages are employed in spoken dialog between Ke Jia and its users. The information in the dialog is extracted, classified and transferred into inner representation by Ke Jia's NLP mechanism, and further used autonomously in problem-solving and planning. A series of case study was conducted on Ke Jia with positive results, verifying its ability of acquiring knowledge through spoken dialog with users, autonomous solving problems by virtue of acquired causal knowledge, and autonomous planning for complex tasks.

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Artificial Intelligence

General Terms

Design, Experimentation

Keywords

Human-robot interaction, Cognitive robotics, Modeling natural language, Knowledge representation

1. INTRODUCTION

Remarkable progress has been made on research into intelligent robots, in particular, service robots. Also in recent years, there has been an increasing interest in integrating techniques drawn from areas of AI and Robotics, including vision, navigation, manipulation, machine learning, planning, reasoning, speech recognition and natural language processing [17, 4, 15, 16, 2, 3, 6, 7].

One of the most attractive ideas from these efforts is human-robot collaboration, according to which robots should not be taken as tools, but rather as our partners [9]. This especially applies to service robots. In the most conceivable application scenarios like offices and homes, humans want service robots help them do a lot of things, which they possess complete mental and physical capabilities of doing these things by themselves. Therefore, humans can help robots in these situations, especially with their knowledge, so that the robots can provide humans with better service, especially in labor work. Since there would be a long period before robots gain human-like intelligence, human-robot collaboration is beneficial and even necessary for many real-world applications in near future, and especially significant for the aging society.

A necessary condition of and crucial means to human-robot collaboration is natural and powerful human-robot communication. This in turn demands powerful ability of commonsense reasoning and planning, so that the knowledge acquired can be made use of by the robots autonomously. Based on the consideration, Ke Jia Project has been launched, trying to develop high-level cognitive functions of service robots suitable for human-robot collaboration. This effort is based on state-of-the-art, general-purpose NLP and commonsense reasoning techniques.

Ke Jia robot has been implemented in upgrade versions and a series of case studies has been carried out. In the first type of case study, Ke Jia was examined with standard tests in RoboCup@home league competitions, such as building a map of an unknown environment, identifying humans, following an unknown person, etc [14]. In the second type of case study, the robot was given complex tasks, each being composed of more than one simpler task, where each simpler task may consist of multiple atomic actions. The experiments showed that Ke Jia can understand the interconnection between the simpler tasks and make optimal plans with the atomic actions. In the third type of case study, Ke Jia was taught some causal knowledge through spoken human-robot dialog. By virtue of the acquired knowledge, the robot succeeded in making "careful" plans whose execution will realize the assigned goals while avoiding unwanted side-effects. All the experiments show that Ke Jia's ability can be substantially raised through human-robot collaboration.

Section 2 explains our motivations for Ke Jia Project in detail. Section 3 presents the framework and some of its features. Ke Jia's main mechanisms, NLP and commonsense reasoning, and their integration are described in Section 4. We report on our case studies in Section 5 and give some conclusions in Section 6.

Cite as: Developing High-level Cognitive Functions for Service Robots, X. Chen, J. Ji, J. Jiang, G. Jin, F. Wang, and J. Xie, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 989-996

2. MOTIVATIONS

We are concerned with application scenarios which conform to following three assumptions. The first assumption is well-accepted in the area for most of applications, while the second and the third one have not been adopted by all researchers explicitly. However, we believe all the three hold for a majority of more complicated real-world applications and play an important role in the relevant research.

(A1) *Common users*: Typically, an intelligent service robot is expected to serve untrained and non-technical users. As a consequence, the robot should be equipped with some intuitive, human-oriented user interface, so that it can be employed by these users without instruction [17, 8, 2, 3]. A huge amount of efforts have been made on this "uppermost" requirement and a variety of techniques for human-robot interaction have been developed, including spoken language recognition, gesture recognition, facial perception, etc [8], sometimes integrated with more traditional techniques such as graphical user interface.

(A2) *Human-robot collaboration*: In many cases, human users need to assist the robot on its missions one way or the other. An example is task learning, where a human teaches a robot how to perform a specific task through a combination of spoken commands, observation and imitation of the human's performing that task [16]. More generally, it is proposed that human users and robot(s) should collaborate to solve problems, where humans assist robot(s) with cognition and perception [9]. More explanations are given in Section 1.

(A3) *Underspecification*: Unlike an industrial robot, the tasks of a service robot are frequently underspecified, ie, not predefined completely, because users usually provide underspecified descriptions about their intentions (eg, tasks) and the environments are typically unpredictable and dynamic [17, 2, 3]. Of course, one can choose to develop service robots of which the tasks are defined completely in advance. But this choice means that the robots have no sufficient capability to response/adapt to their unpredictable and dynamic environments, as well as the users.

Based on these assumptions, we identify four requirements for Ke Jia Project to meet.

(R1) *Ability of acquiring knowledge from users.* A straightforward way is to utilize human-robot dialog as means to acquire a variety of knowledge from users or designers, including descriptions about the environment and the task at hand, even deeper knowledge such as causation related to the tasks.

(R2) *Feasibility of the expression of complex tasks.* In more complicated applications, the robots must be able to handle complex tasks, not only simple ones. For example, "clean the house" is a complex task, while "open the door" is a simple one. A simple task is not necessarily an easy task for robots.

(R3) *Appropriate degrees of autonomy.* Although we do not expect a service robot can do everything by itself, some degree of autonomy is required absolutely. An autonomous robot should be able to make use of acquired knowledge to solve problems, especially, plan for complex tasks.

(R4) *Real-time inference.* A well-known fact in AI and related areas is that more powerful mechanisms in functions are usually more time-consuming in computation. Therefore, it is necessary to take into account this fact when developing more powerful intelligent service robots, since real-time processing is demanded for real-world applications.

An important observation, to our best knowledge, is that *commands recognition* has been the dominate RHI technique in current efforts on intelligent service robots. With this technique, the number of commands and the format of each command are fixed beforehand. In human-robot communication, the robot tries to match each utterance of its users with a predefined command. Once a command is recognized, it is executed by the robot through running a course of low-level commands, which is manually programmed beforehand or produced by the motion planning module according to the command. This technique has shown good performance in many simple applications [3]. For more complicated applications with the requirements above, however, there are still many challenges.

For example, how to express complex tasks? There are two options of specifying complex tasks within the scope of commands recognition. The first one is to use high-level commands, one for each complex task. Generally, the more complicated a task is, the more parameters are contained in the corresponding command. For example, the command "turn <right>" has just one parameter, while the command "move the <red> <bottle> from the <table> to the <teapoy>" has four. Therefore, as there will be more and more high-level commands with more and more parameters, it becomes harder and harder for users to remember and use these commands. Moreover, given a certain sort of task, its instances under different contexts where the task is executed may need different parameters in the task specification. For example, if there is more than one red bottle on the table and the user wants to move a particular one of them, an additional parameter, eg, an attribute which distinguishes between this bottle and the others, has to be introduced into the "move" command above. Obviously, it would be too difficult or even impossible for the designers to specify beforehand all the necessary parameters of each high-level command. In these cases, the "high-level commands" proposal makes infeasible demands on both users and designers, and contradicts Assumption (A1) and (A3).

The other option of employing commands recognition to express complex tasks is *commands combination*, where simple commands, each representing a simple task, are combined to specify a complex task. This way a human user has to choose and arrange in some appropriate order all the simple commands for the complex task at hand, so that the execution of these commands in that order by the robot will fulfill the task. This means that it is the human users, not the robot, who are responsible for task planning; therefore, the robot has a very limited degree of autonomy. However, for many complex tasks, humans may only be able to describe the goal states they want to reach. So the robot should take charge of task planning, as well as motion planning, according to Assumption (A1).

To meet all the requirements under the assumptions, we proposed an alternative approach based on the state-of-the-art NLP and common-sense reasoning techniques, integrated with human-robot dialog and motion planning. The main ideas are described below.

(1) We take some *limited segments of natural language* (LSNLs) as RHI languages. A specific LSNL is formed with a fixed vocabulary and a simplified syntax, a subset of the syntax of some natural language. With these LSNLs, service queries, descriptions about the states of environments, knowledge of the world, instructions about new tasks and so on can be expressed in similar ways as in everyday spoken language dialog and teaching at classes. Accordingly, we employ and develop some NLP techniques for the robot to "understand" the dialog in these LSNLs.

(2) We introduced Answer Set Programming (ASP) as knowledge representation and reasoning tool for Ke Jia. ASP is a logic language with Prolog-like syntax and the stable model semantics, and thus a non-monotonic reasoning mechanism [10]. This feature makes it suitable for handling underspecification and further supporting knowledge accumulation and human-robot collaboration through dialog.

(3) We proposed a layered architecture (Figure 1) to integrate all the techniques and separate the task and motion planning. This is crucial for reducing computational costs, since current ASP solvers are not efficient enough for motion planning.

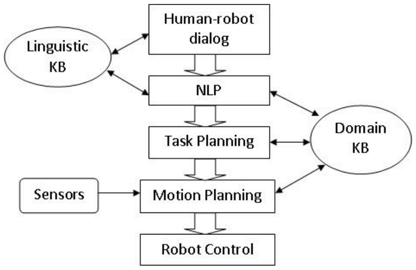

Figure 1: The layered architecture

3. FRAMEWORK AND FEATURES

System Overview. The hardware framework of Ke Jia robot is shown in Fig. 2. Its sensors include a laser range finder, a stereo camera and a set of sonars. The robot has an arm for manipulating portable items. The computational resources consist of a laptop and a on-board PC. It is worthwhile emphasizing that neither additional computational resources off-board nor remote control is needed for the robot when it performs its tasks. This means all the computation is carried out on-board. Similar to RHINO [3] and STAIR [15], distributed and asynchronous processing are adopted, with no centralized clock or a centralized communication module in Ke Jia's system.

The software architecture is shown in Fig. 1. Ke Jia is driven by input from human-robot dialog. The information from the dialog is extracted, classified and transferred into the task planning module through a three-steps procedure of the NLP module. Within the task planning module, the information and knowledge are represented as an ASP program, which may vary from time to time along with the new information from the dialog and the observation. However, there are some "planning states" where the ASP program keeps no change and represents a certain task as well as the related knowledge. Then Ke Jia's task planning module generates an optimal high-level plan for the task, and feeds it to the motion planning module. A low-level plan corresponding to each high-level plan will be generated by the motion planning module and executed by the robot control module.When needed, however, Ke Jia will ask the user for further information and re-plan. As mentioned above, we use LSNLs in the spoken human-robot dialog and thus there are some challenges. Since all the LSNLs we have used are very small, so far there is no substantial obstacle to the effort on our main goals.

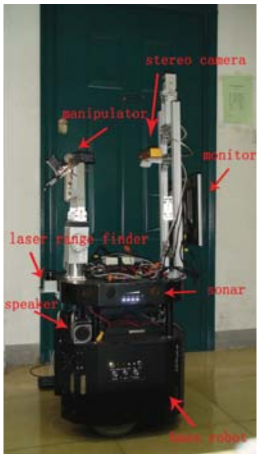

Figure 2: The hardware framework of Ke Jia

Task Planning. There are no well-accepted criteria for the division of task and motion planning and thus the division depends on designing choices. Some of *atomic tasks* we defined in Ke Jia Project are listed in Table 1. Each atomic task, also called an *action*, is designed as a primitive for Ke Jia's task planning and can be handled further by Ke Jia's motion planning. With the specification of atomic tasks, the division between the task and motion planning of Ke Jia is clearly defined.

An outstanding feature of Ke Jia's actions is underspecifiedness, which supports flexibility of representation and communication. In fact, for example, all phrases semantically equivalent to "pick-up an item" in the given LSNL are identified by Ke Jia as instances of this action, although most of the phrases may not specify the action completely so that it can be executed by a robot. Generally, people cannot afford to explicitly spell out every detail of every course of action in every conceivable situation [17]. Suppose, for example, there exist two bottles on the table and a user just wants to get a particular one from them. In this case, the user may express his/her query through a phrase like "bring me the bottle on the table". If Ke Jia finds two bottles on the table during its execution of this task, say, when it drives by the table, it will ask the user to provide further information. There are lots of more complicated phenomena of underspecification about objects and their attributes

action	**function**
goto a location	drive to the assigned location from the current location
pick-up an item	pick-up the assigned item, return “importable” if the item is not portable
put-down at a position	put down the item in hand at the assigned position
search for an object	search for the assigned object through sensors and return the position of the object if succeed

Table 1: Atomic actions

which can be resolved with AI technology.

To realize these features, it is assumed by default in Ke Jia's NLP and task planning module that any singular noun represents a single object. This way, Ke Jia's task planning module can *plan under underspecification*, generating (underspecified) high-level plans and providing Ke Jia with the possibilities of acquiring more (detailed) information if necessary. Once new information reducing uncertainties is received, Ke Jia updates its world model and re-plans if needed. For this purpose, non-monotonic inference is necessary. This is one of the main reasons that Ke Jia employs ASP as its inference tool.

Motion Planning and Robot Control. A set of elementary actions are defined for Ke Jia's motion planning. Each elementary action is pre-defined with a fixed set of parameters, similar to a command in command recognition in some aspects. Unlike commands, however, elementary actions are determined by the capabilities of a robot's hardware to a great extent. An elementary action is full-specified in the sense that if only the values of parameters of an elementary action are assigned, the robot control module will execute the elementary action “blindly”. For instance, once the robot gets the position information of the bottle while performing the elementary action “catch the bottle”, it will try to catch the object in that position, no matter what object it is. In fact, it is not the “responsibility” of robot control module and/or elementary actions to identify the “right” objects for acting.

For each atomic action in a high-level plan generated by Ke Jia's task planning module, the motion planning module will try to make a low-level plan composed of elementary actions, and execute the low-level plan autonomously. This is a special form of hierarchical planning introduced in Ke Jia for gaining its computational efficiency. Currently, we employ heuristic methods for Ke Jia's motion planning due to the same reason.

World Model. Logically, the set of objects for Ke Jia's planning is determined by the specific LSNL, containing all the individuals expressible with noun phrases of the LSNL. However, most of these individuals may not exist in the environment. On the contrary, only those objects actually perceived by the robot constitute the domain of objects. Moreover, the robot may perceive more objects in the environment during the execution of a task. In order to capture the changing domain of actual objects and other “real” attributes of the environment, Ke Jia maintains a world model (as a part of domain KB), which is shared by the NLP, task planning and motion planning module and can be updated with new information from the human-robot dialog and/or observation. These modules coordinate their behaviors through the shared knowledge and information. Therefore, when Ke Jia acquires new information that there are two bottles, one red and one green, on the table, it will update its world model and ask the user to indicate which bottle he/she wants to get.

4. COUPLING NLP WITH ASP

This section describes the NLP and ASP based commonsense reasoning techniques employed by or developed for Ke Jia. Generally, these two techniques are studied or applied separately. We coupled them in Ke Jia system.

For each input sentence, the NLP module works in three steps: (1) Parsing, in which input sentence is parsed with the Stanford parser [12]; (2) Semantic analysis, in which labeled logic predicates are generated to represent the meaning of the input sentence; (3) Pragmatic analysis, in which the NLP module detects the linguistic function of the input sentence and rewrites the labeled logic predicates into unlabeled ones, so that they are recognizable by ASP.

A single input of the NLP module from the human-robot dialog module is a string of words, which is regarded as a sentence. It is parsed by the Stanford parser, which returns two kinds of information on the syntactic structure of the sentence: a grammar tree after the UPenn tagging style, and a set of typed dependencies between all the words in the sentence [5].

The building of semantic representation is mainly based on typed dependencies, though sometimes the syntactic categories of words and phrases are required, too. The semantic representation is a set of semantic elements. A semantic element can be either a first order predicate, tagged with a label, or a variable tagged with a label. The arguments of first order predicates are labels, so a predicate can be in the argument list of another predicate. Thus, some second order information can be expressed in first order settings, such as “a verb predicate is modified by an adverb predicate”. This approach follows Segmented Discourse Representation Theory(SDRT) [1]. Each word corresponds with a semantic element, and each entity in the semantic space has its own semantic element, too. Semantic elements are divided into five types: modifier, entity, verb, preposition and conjunction.

Modifier elements are used to represent nouns, pronouns, adjectives, adverbs and all words that behave as an NP by itself. They are either standalone or not standalone. A standalone modifier element always carries its own entity element. Usually, they are derived from nouns or pronouns. A non-standalone modifier element must have a label from somewhere else to be filled in its argument list. Each modifier element has only one argument. Entity elements do not correspond to any word in the text. They do not have any arguments, and would not be finally printed as predicates, but as variables. Verb elements corresponds to verbs, and have 1-3 arguments to represent its subject, direct object and indirect object, if there are any; preposition elements corresponds to prepositions, and have exactly 2 arguments, represent its subject and object, respectively; conjunction elements correspond to conjunction words, and can have 2 or more arguments, representing all its conjuncts.

Each type of semantic elements has different properties and behaviors. With these definitions, we can deal with all natural language grammatical phenomena that are specified by the typed dependencies. The typed dependencies specify the relations between the words in a sentence. Because each word corresponds to a semantic element, so the relations between semantic elements are also specified by the typed dependencies. According to these relation specifications, the arguments of each semantic element can be determined.

For example, in an amod (adjective modifier) dependency, the dependent word modifies the governor word, and the two words are both modifier elements. For every amod dependency, its dependent element and governor element should have same subject. Thus we can assign the subject argument of the governor element to the subject argument of the dependent element. For each dependency, we have defined how to fill the unfilled argument list according to the type of the dependency. When all the typed dependencies of the sentence are dealt with, each semantic element will correctly represent the entities, properties of the entities and relations among the entities specified by the sentence. Thus the meaning of the sentence is captured in logical forms.

Now pragmatic analysis is launched to transform the above logical forms into ASP programs. When a prepositional phrase modifies a verb, it should be combined with the verb, producing a composite predicate, and eliminate any reference to another predicate from the argument list of a predicate, so that the result is purely of first order. In addition, there are more subtle issues related to the words corresponding to so-called logic connectives. We will return to these issues after a brief introduction of ASP.

ASP is proposed in [10]. An ASP program is a finite set of *rules* of the form:

$$H \leftarrow p_1, \ldots, p_k, not\, q_1, \ldots not\, q_m, \tag{1}$$

where p_i, $1 \leq i \leq k$, and q_j, $1 \leq j \leq m$, are literals, and H is either empty or an literal. A *literal* is a formula of the form a or $\neg a$, where a is an atom. If H is empty, then this rule is also called a *constraint*. A rule consisting of only H is called a *fact*. There are two kinds of negation in ASP, the classical negation $\neg$ and non-classical negation *not*, ie, *negation as failure*. The meaning of *not a* can be explained as saying that "a is not derivable from the program". Similarly, a constraint $\leftarrow p_1, \ldots, p_k$ specifies that $p_1, \ldots, p_k$ cannot be derived jointly from the program.

Commonsense knowledge can be represented easily with ASP programs. For example, Ke Jia's ability of 'catch' and the corresponding predicates *hold* and *empty* can be expressed as follows:

$$\begin{aligned}
catch(A,T) :- \;& not\, n\_catch(A,T), small\_object(A),\\
& time(T), T < lasttime.\\
n\_catch(A,T) :- \;& not\, catch(A,T), small\_object(A),\\
& time(T), T < lasttime.\\
hold(A,T+1) :- \;& catch(A,T), small\_object(A), time(T),\\
& T < lasttime.\\
n\_catch(A,T) :- \;& location(A,X,T), small\_object(A),\\
& not\, location(agent,X,T), number(X), time(T).\\
n\_catch(A,T) :- \;& not\, empty(T), small\_object(A),\\
& number(X), time(T).
\end{aligned}$$

$$\begin{aligned}
empty(T+1) :- \;& empty(T), not\, n\_empty(T+1),\\
& time(T), T < lasttime.\\
n\_empty(T+1) :- \;& hold(A,T+1), small\_object(A),\\
& time(T), T < lasttime.\\
hold(A,T+1) :- \;& hold(A,T), small\_object(A), time(T),\\
& not\, n\_hold(A,T+1), T < lasttime.\\
n\_hold(A,T+1) :- \;& empty(T+1), small\_object(A),\\
& time(T), T < lasttime.
\end{aligned}$$

The first two rules state that, at any time T, either $catch(A,T)$ or $\neg catch(A,T)$ hold, but not both because of the classical negation ($\neg catch(A;T)$ is written as $n\_catch(A;T)$ in above program). The third rule states the *effect* of the action *catch*, if the robot catches an object at time T, then she holds the object at time $T+1$. The next two rules state the inexecutable conditions for *catch*, if the robot is not at the same position with the object or the hand of the robot is not empty, then she cannot catch the object. The last four rules are *inertia* rules for predicates *hold* and *empty*, which concern the frame problem.

Now return to the last step of the NLP module, pragmatic analysis. So far in Ke Jia Project, we have considered two words corresponding to two most important logic connectives, "if" and "not". The word "if" has more "powerful" functions than a mere predicate. It can function as a variety of entailment, such as logical implication, counterfactual conditional or causal entailment. Since our concerns (human-robot interaction and collaboration, robot planning, etc) are mainly related to commonsense reasoning and causal entailment, we rewrite conditionals (sentences of the form "if-then") to ASP rules. This implies that any conditional in our LSNLs is understood by Ke Jia as a *default*. Similarly, word "not" is also understood as a commonsense term. There are three cases where "not" is permitted to appear in current LSNL sentences. (i) "not" is used to form a negative, imperative sentence, such as "do not open the door". The whole sentence expresses that something is forbidden and thus should be translated naturally into an ASP constraint. (ii) "not" modifies the main verb of a sentence or clause. These sentences or clauses should be handled similarly to a negative, imperative sentence as above. In particular, negative if-clauses are understood as defeasible conditions and thus rewritten as ASP rules. (iii) "not" modifies "anything", representing "nothing". The sentence or clause should be translated into an ASP rule too. In all the cases, word "not" is translated naturally or approximately into the negation-as-failure operator, *not*. Most of other usages of "not", say, modifying nouns or adjectives, should be translated into classical negation. We have not handled these cases in Ke Jia Project, since more work is needed to support the implementation. After the whole procedure of Ke Jia's NLP module, all the sentences from the LSNLs can transformed into ASP programs, so that the information and knowledge expressed in these sentences can be utilized by the task planning module.

ASP provides a unified mechanism of handling commonsense reasoning and planning with a solution to the frame problem. A lot of ASP solvers have been developed too. One can solve an ASP program by running it on an ASP solver. The outcomes are called answer sets, each containing a set of literals which can be derived jointly from the program, or intuitively, hold jointly under the program as a KB. In particular, an answer set is actually a plan if the program specifies a planning problem. For instance, the rules listed above come from a program specifying a planning problem. For sake of efficiency, we actually employed more advanced forms of ASP, say *action languages* $\mathcal{C}+$ [11], to represent the knowledge and specify the planning problems of Ke Jia.

5. CASE STUDY

We have conducted several types of case study in the efforts on Ke Jia Project. In Case Study 1, we took standard tests in RoboCup@home league competitions as benchmarks, including building a map of an unknown environment, identifying humans, following an unknown person through a dynamical environment, etc [14]. The aim of this case study is to verify Ke Jia's "basic capabilities". Actually, however, it is not necessary for a service robot to pass these tests by using the main technical contributions described in this paper. Therefore, we will not present this type of case study here. But we believe that the techniques reported in this paper will be significant or even necessary for some new tests or new versions of the current tests in the future. In this section, we will describe the second and third type of case study, where we took complex tasks and causal reasoning tasks as benchmark, respectively.

5.1 Case Study 2: Complex Tasks

A complex task is composed of more than one simple task. The more powerful the ability of task planning of a robot is, the better performance for complex tasks the robot will have. Without this ability, some complex tasks such as "clean the house" cannot be realized by the robot. In simpler cases, the robot with poor task planning ability cannot carry out complex tasks optimally.

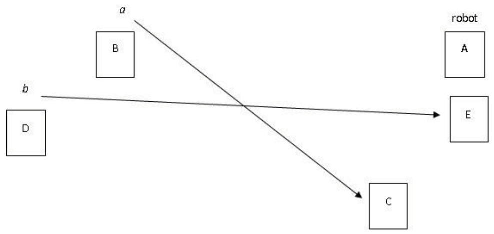

Figure 3: The initial and the goal state of the complex task

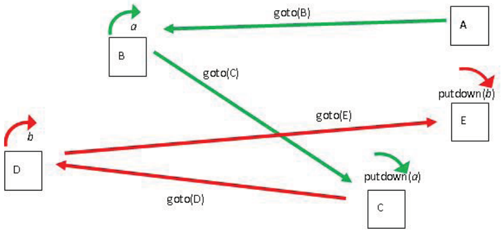

Figure 4: Perform two component tasks separately

The complex tasks we chose in this case study are service queries of following form: "move a from position B to C and move b from D to E", given the initial state is shown in Fig. 3, where the robot is at location A and portable objects a and b are in position B and D, respectively. Note that the two component tasks, "move a from position B to C" and "move b from D to E", are not even Ke Jia's atomic actions. This increases the complexity of the complex task.

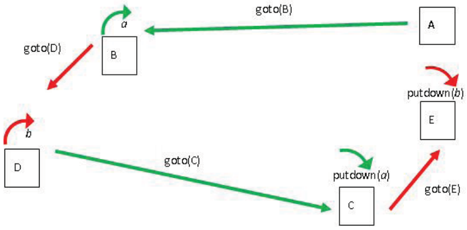

Figure 5: Perform the complex task optimally

As a comparison, consider how this kind of tasks will be fulfilled by a service robot that employs commands recognition/combination technique, provided that it can complete it anyway. Since such a robot only uses command language in HRI, we assume that the two component tasks be instances of a command, "move x from y to z". Obviously, by using commands recognition/combination technique, these two commands will be performed successively and separately, as shown in Figure 4. Generally this is not an optimal solution to the panning problem.

Ke Jia takes the user request as a single complex task and makes a plan interleaving the execution of the two component tasks this way: *goto*(B), *pickup*(a), *goto*(D), *pickup*(b), *goto*(C), *putdown*(a), *goto*(E), *putdown*(b), as shown in Figure 5. This plan is optimal with respect to the cost of Ke Jia's atomic actions, which comprises the distances between the locations/positions. So this is a most efficient solution to the panning problem. More importantly, it is worthwhile pointing out that this plan was made autonomously and completely with the framework and mechanisms described in Section 3 and 4. No matter whatever the locations/positions are chosen in the environment, Ke Jia is always able to generate an optimal plan for this complex task. This also means that Ke Jia can understand the interconnection among the atomic tasks contained in the complex task through its NLP mechanism. These features benefit from Ke Jia's general-purpose mechanism of natural language understanding and commonsense reasoning.

In the experiments, Ke Jia was given the task in more natural forms, such as an English sentence within the LSNL like "give me the green bottle and put the red bottle on the table." The experimental environment is a simplified home environment[1]. The real-time performance for completing this complex task is also acceptable. Generally, it took about 0.8 second for Ke Jia to accomplish the task panning. The computation of other modules is less time-consuming.

In the experiments, we also tested Ke Jia's ability of acquiring simple knowledge from users through spoken dialog, in order to reduce uncertainty about the task at hand due to the underspecifiedness of task description. At the beginning of the above task, Ke Jia did not know the position of the bottles and asked the user to provide relevant information, although it could find them with its search function. Once the user had answered the questions, Ke Jia fulfilled the task

[1] We made two demos for this case study, see `http://wrighteagle.org/media/task_comp_090627.mpg` and `http://wrighteagle.org/media/complex_090910.mpg`.

successfully. This result shows that Ke Jia can understand and make use of users' description about the environment, such as "the green bottle is on the teapoy."

5.2 Case Study 3: Causal Reasoning

We tried to test and verify Ke Jia's ability of handling more complicated and difficult problems. One sort of problems we chose involves causal reasoning. It is well-known that the capability of reasoning with causation is very important and even crucial for many real-world applications and there has been a lot of theoretical research on causal reasoning. However, though there is work on training robots, eg, teaching a robot how to perform a specific task through a combination of spoken commands and observation [16], there is no work on complementary ways to robot training, eg, teaching robots causal knowledge only through spoken dialog so that the robot gains better capability immediately. This case study aims at both goals at the same time.

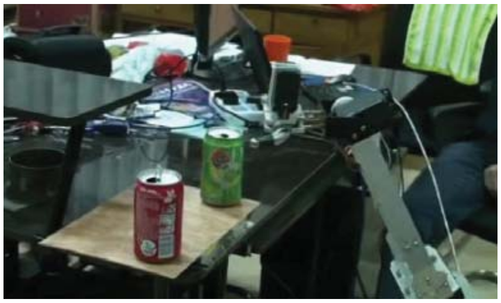

Figure 6: Setting of Case Study 3

The causal reasoning problem we used in this case study is related to commonsense knowledge about "balance" and "fall". A testing instance is shown in Figure 6. There is a board putting on the edge of a table, with one end sticking-out. A red bottle is put on the sticking-out end of the board, and a green bottle is on the other end. If the green bottle was moved while the red one was on the sticking-out end of the board, the red bottle would drop to the ground. The task is to pick up the green bottle under a default presupposition of avoiding anything falling. To accomplish the task, one must take some measure first against the unwanted side-effect of action "catch the green bottle".

Obviously, causal reasoning is needed for making an appropriate plan for the task. Moreover, currently no robots could extract the causal knowledge from observation on the setting, and thus acquiring the relevant knowledge is necessary. One of ways we have been trying is to teach Ke Jia the causal knowledge through spoken human-robot dialog. This task is more complicated and difficult than the complex task tested in Case Study 2, in the sense that causal knowledge is deeper than the phenomenal descriptions of environments.

We tried two ways in Case Study 3. (1) The first one is to input into Ke Jia some causal knowledge which is manually programmed in ASP rules, and provide some facts needed for making use of the causal knowledge. Only four rules are needed for the specification of a relatively simple notion of "balance" and "fall" (see Fig. 7). The literal $holds(falling(A), t)$, where $holds$ is a meta-predicate, means that the proposition that object A falls at time point t holds. Intuitively, the first rule states that a long-shape object will keep balance if there is something on each end of it. The second one states that one end that is not sticking-out is steady. The third and fourth one state that any items on the sticking-out end would fall if there was nothing on the other, steady end.

Using the build-in causal knowledge, Ke Jia accomplished the task with some factual information about the task environment. Ke Jia got the additional information through spoken dialog with the user, where it was told that the red bottle is on the sticking-out end of the board and the green one is on the other end[2]. Ke Jia produced a plan in which the red bottle is moved to a safety place first and then the green bottle is caught. The inference for generating the plan costs no more than 0.5 second in the experiments. The results demonstrate that Ke Jia can make use of build-in causal knowledge to accomplish causal reasoning easily. This also implies a possibility of inputting into service robots commonsense knowledge from large-scale knowledge systems such as Cyc [13].

(2) The second way of equipping Ke Jia with causal knowledge is to teach causal knowledge directly through spoken human-robot dialog, instead of inputting manually programmed ASP rules. For sake of simplicity, we taught Ke Jia simpler causal knowledge about "fall", where deeper causal knowledge about "balance" hides behind. Although the causal knowledge taught this way is simpler, this second way of Case Study 3 provides stronger and clearer evidence about Ke Jia's ability of natural language understanding and causal reasoning.

In the experiments, we took a version of Ke Jia without any build-in knowledge about "balance", "fall", or any other equivalents. Ke Jia was told in the dialog that an object will fall if it is on the sticking-out end of a board and there is nothing on the other end of the board. With its NLP mechanism, Ke Jia extracted the knowledge and transformed it into ASP rule

$$\begin{aligned} holds(falling(A), t) \leftarrow\ & sticking\_out(D), holds(on(A, D), t),\\ & endof(D, B), board(B), endof(E, B),\\ & not\, holds(on(C, E), t). \end{aligned}$$

Then the ASP program specifying the task at hand was updated by adding this rule into it. Ke Jia was also told that the red bottle was on the sticking-out end of the board and the green one was on the other end of the board. Pieces of the factual information were also extracted and transformed into the ASP program. With the knowledge and information, Ke Jia accomplished the task with the same plan: moving the red bottle first and them catching the green bottle.It is worthwhile emphasizing the fact that Ke Jia could not accomplish the task without the causal knowledge acquired through the dialog. This indicates that Ke Jia's ability is substantially raised by knowledge acquisition through dialog.

For both ways of Case Study 3, using build-in knowledge and taught by humans, it took no more than 0.5 second for Ke Jia to generate the plan.

6. CONCLUSIONS

The primary goal of Ke Jia Project is to establish mechanisms on a unified framework suitable for human-robot collaboration for service robots. We take human-robot collaboration as an assumption, rather than as just a require-

[2] See the demo at `http://wrighteagle.org/en/demo`.

$$\mathit{holds}(\mathit{balance}(B,A,C),0) \leftarrow \mathit{holds}(\mathit{on}(A,E_1),0), \mathit{holds}(\mathit{on}(C,E_2),0), \mathit{endof}(E_1,B), \mathit{endof}(E_2,B).$$

$$\mathit{holds}(\mathit{steady}(A),0) \leftarrow \mathit{holds}(\mathit{on}(A,E),0), \mathit{not}\,\mathit{sticking\_out}(E).$$

$$\mathit{holds}(\mathit{falling}(A),t) \leftarrow \mathit{holds}(\mathit{on}(A,B),t), \mathit{holds}(\mathit{balance}(B,A,C),t-1), \mathit{not}\,\mathit{holds}(\mathit{on}(C,B),t), \mathit{not}\,\mathit{holds}(\mathit{steady}(A),t).$$

$$\mathit{holds}(\mathit{falling}(C),t) \leftarrow \mathit{holds}(\mathit{on}(C,B),t), \mathit{holds}(\mathit{balance}(B,A,C),t-1), \mathit{not}\,\mathit{holds}(\mathit{on}(A,B),t), \mathit{not}\,\mathit{holds}(\mathit{steady}(C),t).$$

Figure 7: Rules for "balance" and "fall"

ment. Two more assumptions, common users and underspecification, are adopted and accordingly four requirements are identified for Ke Jia robot to meet. We employ state-of-the-art NLP and commonsense reasoning techniques as basic mechanisms for human-robot communication and task planning. A series of case studies was conducted with positive results, verifying Ke Jia's ability of acquiring knowledge through spoken dialog with users, autonomous solving problems by virtue of acquired causal knowledge, and autonomous planning for complex tasks.

To a great extent, in this paper we could only report on results we have gotten so far in Ke Jia Project. Actually, there are a lot of challenges in the efforts, most of which are still under investigation. For example, computational efficiency of task planning has been a crucial issue, although we succeeded in making Ke Jia's computation faster and faster. However, we are optimistic about this issue, since there has been an increasing interest in developing more and more efficient ASP solvers, which will provide us with an "additional" source to attack the problem. It is the case for issues in NLP. A challenge particular to us is in the coupling of NLP and ASP, ie, establishing a general-purpose mechanism of transforming limited segments of natural languages into action languages. We believe that Ke Jia Project will benefit from all the challenges.

7. ACKNOWLEDGMENTS

This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundations of China under grant 60745002. We thank Fengzhen Lin, Daniele Nardi, Yan Zhang, and Shlomo Zilberrstein for helpful discussions about this effort. We are also grateful to the anonymous reviewers for their constructive comments.

8. REFERENCES

[1] N. Asher and A. Lascarides. *Logics of conversation*. Cambridge University Press, 2003.

[2] H. Asoh, N. Vlassis, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Ito, T. Kurita, T. Matsui, R. Bunschoten, and B. Kröse. Jijo-2: An office robot that communicates and learns. *IEEE Intelligent Systems*, 16(5):46–55, 2001.

[3] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[4] X. Chen, J. Jiang, J. Ji, G. Jin, and F. Wang. Integrating nlp with reasoning about actions for autonomous agents communicating with humans. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-09)*, pages 137–140, 2009.

[5] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of Fifth International Conference on Language Resources and Evaluation (LREC-06)*, pages 449–454, 2006.

[6] P. Doherty, J. Kvarnström, and F. Heintz. A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 19(3):332–377, 2009.

[7] A. Ferrein and G. Lakemeyer. Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*, 56(11):980–991, 2008.

[8] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 42(3-4):143–166, 2003.

[9] T. Fong, C. Thorpe, and C. Baur. Robot, asker of questions. *Robotics and Autonomous Systems*, 42(3-4):235–243, 2003.

[10] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth International Conference on Logic Programming (ICLP-88)*, pages 1070–1080, 1988.

[11] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, H. Turner, and J. L. V. Lifschitz. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):2004, 2004.

[12] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems (NIPS)*, 15:3–10, 2003.

[13] D. B. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[14] D. Nardi, J.-D. Dessimoz, P. F. Dominey, L. Iocchi, J. R. del Solar, P. E. Rybski, J. Savage, S. Schiffer, K. Sugiura, T. Wisspeintner, T. van der Zant, and A. Yazdani. Robocup@home: Rules and regulation, 2009.

[15] M. Quigley and A. Y. Ng. STAIR: hardware and software architecture. In *AAAI 2007 Robotics Workshop*, 2007.

[16] P. E. Rybski, K. Yoon, J. Stolarz, and M. M. Veloso. Interactive robot task training through dialog and demonstration. In *Proceedings of the Second ACM/IEEE international conference on Human-robot interaction (HRI-07)*, pages 49–56. ACM, 2007.

[17] M. Tenorth and M. Beetz. KnowRob — knowledge processing for autonomous personal robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-09)*, pages 4261–4266, 2009.