Toward Open Knowledge Enabling for Human-Robot Interaction

Xiaoping Chen, Jiongkun Xie, Jianmin Ji, and Zhiqiang Sui
Computer School, University of Science and Technology of China

This paper presents an effort to enable robots to utilize open-source knowledge resources autonomously for human-robot interaction. The main challenges include how to extract knowledge in semi-structured and unstructured natural languages, how to make use of multiple types of knowledge in decision making, and how to identify the knowledge that is missing. A set of techniques for multi-mode natural language processing, integrated decision making, and open knowledge searching is proposed. The OK-KeJia robot prototype is implemented and evaluated, with special attention to two tests on 11,615 user tasks and 467 user desires. The experiments show that the overall performance improves remarkably due to the use of appropriate open knowledge.

Keywords: Human-robot interaction, open knowledge, NLP, decision making, social robotics

1. Introduction

Human-robot interaction (HRI) draws more and more interest from researchers (Burgard et al., 1999; Cantrell et al., 2012; Chen et al., 2010; Doshi & Roy, 2007; Fong, Thorpe, & Baur, 2003; Kaupp, Makarenko, & Durrant-Whyte, 2010; Rosenthal, Veloso, & Dey, 2011; Tenorth & Beetz, 2009; Thrun, 2004). In HRI settings, robots need to be able to communicate with users, understand users' requests, and provide services for users accordingly by taking physical and other actions. For these purposes, a robot needs a lot of knowledge. For instance, when a user tells a robot, "I am thirsty," the robot is expected to do something to meet the user's desire. If the robot knows that this desire may be met by serving a drink to the user, then it can plan and execute actions towards this goal. In many cases, however, it is too hard to equip a robot with complete knowledge before it is put to use. In these cases, the challenge is to develop robots that can acquire ideally automatically missing knowledge from somewhere to accomplish the tasks requested by users at running time.

There are various open-source knowledge resources available on the web, such as Cyc[1], the Open Mind Indoor Common Sense (OMICS) database (Gupta & Kochenderfer, 2004), ontologies, and household appliance manuals. Robots can also gain knowledge through human-robot dialogue. In this paper, we call the knowledge from these resources *open knowledge* and report an effort to acquire and make use of online open knowledge for HRI. We consider three requirements in

[1] http://www.opencyc.org/

Journal of Human-Robot Interaction, Vol. 1, No. 2, 2012, Pages 100–117. DOI 10.5898/JHRI.1.2.Chen

this effort: (i) A robot should be able to understand knowledge in natural language, since a great proportion of open knowledge is expressed in natural language. (ii) A robot should be capable of using different types of knowledge, because a single user task may involve multiple types of knowledge that exist in more than one open-source knowledge resource. (iii) A robot should be able to recognize what gaps exist between the knowledge it already possesses and the task at hand when it encounters knowledge shortages and be able to search for the relevant knowledge from open-source resources.

In the KeJia project, we have made a continual effort to develop intelligent service robots that can meet these requirements. The main ideas are sketched below. First, we develop multi-mode natural language processing (NLP) techniques for comprehension and extraction of knowledge in different modes of natural language expression and for transformation of the extracted knowledge into an intermediate representation. Second, we propose an integrated decision-making mechanism based on a uniform representation of knowledge, so that different types of knowledge can be made use of. Third, we put forth a principle for detecting knowledge gaps between the current task and the local knowledge of a robot, and introduce mechanisms for searching for missing knowledge from open-source resources.

There are projects that share some common concerns, but they do not address all of the aspects mentioned above. KnowRob (Lemaignan, Ros, Sisbot, Alami, & Beetz, 2012; Tenorth & Beetz, 2009) also uses open knowledge such as Cyc, ontology, and OMICS in HRI. The authors study how extraction, representation, and use of knowledge can enable a grounded and shared model of the world suitable for later high-level tasks such as dialogue understanding. A specialized symbolic knowledge representation system based on Description Logics is employed. While their approach is action-centric, where a robot collects and reasons about knowledge around action models, our approach is open-knowledge-centric, focusing on automatically acquiring and utilizing open knowledge for online planning with general-purpose decision-making mechanisms. Cantrell et al. (2012) have done work similar to ours in that open knowledge in natural language is formalized in order to update the planner model and enhance the planning capability of a robot. The major difference lies in the resources and scale of open knowledge, since we use large-scale knowledge resources (e.g., OMICS). In addition, our formalization of open knowledge concerns both unstructured and semi-structured natural language expressions. Rosenthal, Biswas, & Veloso (2010) propose a symbiotic relationship between robots and humans, where the robots and humans benefit each other by requesting and receiving help on actions they could not perform alone due to ability limitations. The Cognitive Systems for Cognitive Assistants (CoSy) project makes a continual contribution to the human-robot dialogue processing for HRI. Kruijff et al. (2010) postulate a bi-directionality hypothesis which relates the linguistic processing to the robots experiences associated with a situated context. They show how such bi-directionality between language and perception influences the situated dialogue processing for HRI. Like CoSy, the situated dialogue processing for HRI is also needed in our system, which provides our robot a foundation for understanding the requests from users and searching for open knowledge. The integrated decision-making mechanism proposed here is also similar to Golog (Levesque, Reiter, Lesperance, Lin, & Scherl, 1997). However, missing steps in a sequence as constraint are allowed and can be filled in by our mechanism, but not in Golog. In fact, Golog programs are intended to be high-level control programs of robots written by the designer, while our approach aims to make robots work by user requests and open knowledge. Talamadupula, Benton, Kambhampati, Schermerhorn, & Scheutz (2010) try to adapt planning technology to Urban Search And Rescue (USAR) with a human-robot team, paying special attention to enabling existing planners, which work under closed-world assumptions, to cope with the open worlds in USAR scenarios. We try to adapt planning technology to HRI, especially by using open knowledge.

In Section 2, we present the framework of the OK-KeJia robot and describe the main ideas of the project. The implementing techniques for two main modules of the robot, multi-mode NLP and integrated decision making, are addressed in Sections 3 and 4, respectively. Some case studies are reported in Section 5, and conclusions are given in Section 6.

2. System Overview

To describe the framework of OK-KeJia robots, we adopt a simple and general language for knowledge representation in this paper. The vocabulary of the language includes objects, predicates and action names. A predicate expresses an attribute of an object, environment or human. For instance, $small(obj)$ expresses that object obj is small. A predicate or its negation is called a literal.

We assume that a robot is equipped with a set of primitive actions. A *primitive action* (action for short) is specified as an ordered pair $\langle pre\text{-}cond(a), \mathit{eff}(a)\rangle$, called an action description, where a is a action name, $pre\text{-}cond(a)$ and $\mathit{eff}(a)$ are sets of literals. Intuitively, $\mathit{eff}(a)$ refers to the effects of executing action a and $pre\text{-}cond(a)$ is the pre-conditions under which a can be executed and $\mathit{eff}(a)$ can be accomplished through execution of a. For instance, $pre\text{-}cond(move(l)) = \{nav\text{-}target(l), \neg robot\text{-}at\text{-}loc(l)\}$ and $\mathit{eff}(move(l)) = \{robot\text{-}at\text{-}loc(l)\}$. The set of all the action descriptions is called the *action model* of the robot and is taken as its built-in knowledge. We require that every action a be implemented by a routine on a real robot, with $\langle pre\text{-}cond(a), \mathit{eff}(a)\rangle$ being the expected action model.

In HRI settings, an action model may be insufficient for a robot to provide certain services to users, since user requests may contain predicates that do not appear in the action model. For example, “thirsty” does not generally appear in the action model of a robot. In this paper, therefore, we consider three further types of knowledge: (i) *Conceptual Knowledge*; that is, knowledge about relationships between concepts. Typically, ontologies specify such relationships, like super and equivalent classes of concepts. (ii) *Procedural Knowledge*; that is, knowledge describing the steps of how to accomplish a task (relationship between a task and its sub-tasks). (iii) *Functional Knowledge*; that is, knowledge about effects of tasks, sometimes called goals. For instance, the effects of task “*put object 1 on table 2*” can be expressed as a set of literals, $\{on(object_1, table_2), empty(grip)\}$. Manual instructions include functional knowledge; for example, the functions of buttons on a microwave oven.

Theoretically, a *growing model* is defined as $M = \langle A, C^*, P^*, F^*\rangle$, where A is the action model and C^*, P^*, and F^* store the conceptual, procedural, and functional knowledge that the robot has obtained, respectively. We assume in this paper that an element of C^* maps a concept onto its superclass/subclass concepts, P^* a task onto its sub-tasks, and F^* a task onto its effects, respectively. C^*, P^*, and F^* can expand during the robot's running. The model provides a framework for analyzing the main research issues. In particular, the integrated decision-making mechanism should be so constructed that knowledge from C^*, P^*, and F^* can be made use of by it. Moreover, knowledge gaps can be identified against the differences between a user task and the local knowledge in M.

The overall architecture of our robot, OK-KeJia, is shown in Figure 1. The robot is driven by input from human-robot dialogue. The information extracted from the dialogue is further processed by the multi-mode NLP module. The integrated decision-making module tries to generate a plan for the current user task. If it succeeds, the plan, consisting of primitive actions, is fed to the low-level control module to execute. The robot tries to acquire open knowledge when it detects knowledge gaps for the current task. The open knowledge searching module is triggered to obtain relevant pieces of knowledge from open-source knowledge resources. A meta-control mechanism (Chen, Sui, & Ji, 2012) is also needed for the coordination of these modules but is not shown in Figure 1. The robot's sensors include a laser range finder, a stereo camera, and a 2D camera. The robot has an arm for manipulating portable items. The on-board computational resource consists of two laptops.

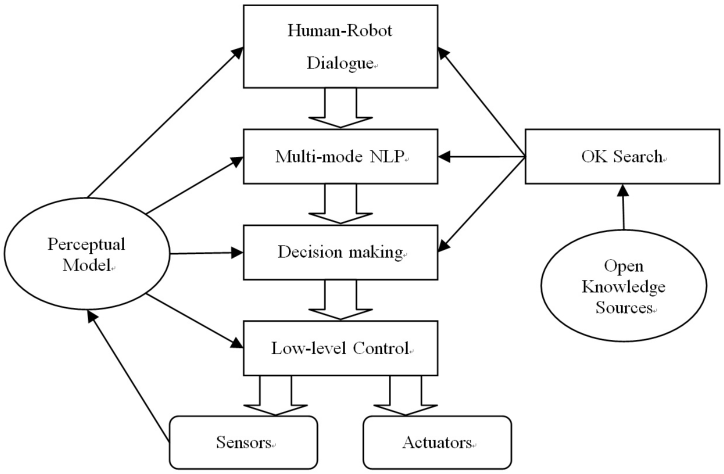

Figure 1. The overall architecture of OK-KeJia

The human-robot dialogue (HRD) component provides the interface for communication between users and the robot. The Speech Application Programming Interface (SAPI) developed by Microsoft is used for speech recognition and synthesis. Once a user's utterance is captured by the recognizer, it is converted into a sequence of words. The embedded dialogue manager (Figure 2) then classifies the dialogue contribution of the input utterance by keeping track of the dialogue moves of the user. In accordance with the dialogue moves, the HRD component decides to update the world model, which contains the information from the perceptual model and of the robot's internal state, and/or to invoke the decision-making module for the task planning, with the semantic representation of the input utterance produced by the multi-mode NLP module, which can process both unstructured and semi-structured natural language expressions (Section 3). At present, the structure of the dialogue is represented as a finite state transition network. Figure 2 shows our implementation (i.e., finite state machine) of managing a simple human-robot dialogue in which the user tells the robot facts that he/she has observed or tasks, and the robot asks for more information if needed. A mixed-initiative dialogue management will be developed in our future work.

Assume that a robot's perception of the current environmental and internal state is expressed as a set of literals, called an *observation*. User tasks are transformed into (dummy, sometimes) goals. Given an observation o and a goal g, a plan for $\langle o, g\rangle$ is defined as a sequence $\langle o, a_1, \ldots, a_n, g\rangle$, where $a_1, \ldots, a_n$ are primitive actions such that the goal g will be reached after the execution of $a_1, \ldots, a_n$ under any initial state satisfying o. In the literature, there are two basic schemas of decision making for autonomous agents and robots that can be employed in open knowledge settings. One is goal-directed planning, the procedure of generating a plan $\langle o, a_1, \ldots, a_n, g\rangle$ for any $\langle o, g\rangle$ with functional knowledge. The other schema is task-directed action selection. This schema employs procedural knowledge iteratively, until the task is decomposed into a sequence of primitive actions. In Section 4, we present a hybrid schema that integrates some mechanisms of task decomposition into a goal-directed planning system.

The ability of a robot to acquire open knowledge depends on the detection of knowledge gaps between the current task and the robot's local knowledge. Let $M = \langle A, C^*, P^*, F^*\rangle$ be the growing model of the robot and $\mathit{eff}(A) = \{p | p \in \mathit{eff}(a)$ for some $a \in A\}$ be the union of all $\mathit{eff}(a)$ where a is a primitive action in A. A predicate Q is *grounded in* M if the following conditions hold: (i)

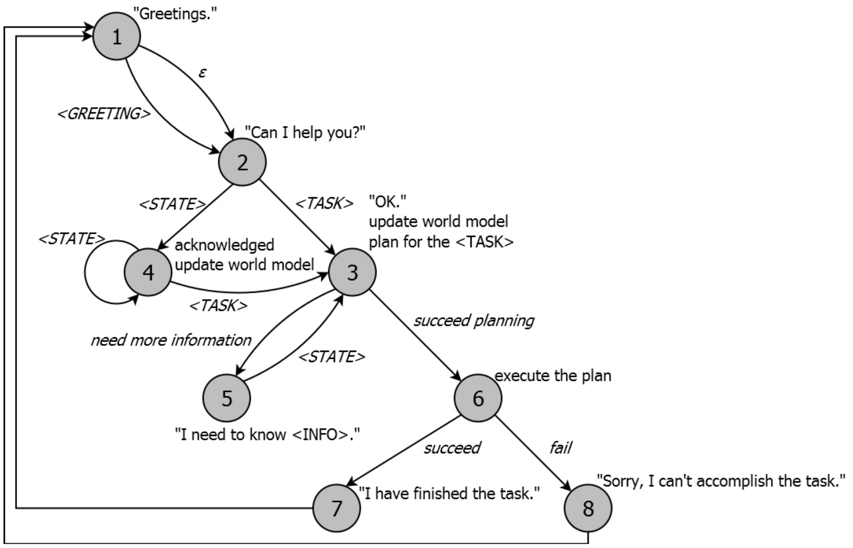

Figure 2. The finite state machine for a simple human-robot dialogue

$Q \in \mathit{eff}(A)$; or (ii) Q is "reduced" (see Section 4) to $Q_1, \ldots, Q_n$ by $C^* \cup P^* \cup F^*$ such that each Q_i $(i = 1, \ldots, n)$ is grounded in M. A task is grounded in M if every predicate in the description of the task is grounded in M. This leads to the definition of knowledge gaps: There is a knowledge gap between a user task p and the robot's growing model M if and only if there is a predicate Q in p such that Q is not grounded in M. Now we can describe the principle of open knowledge searching as follows: Given a task p and a growing model M such that there is a knowledge gap between p and M, search open-source knowledge resources to find C^+, P^+ and/or F^+ (i.e., new knowledge) so that there is no knowledge gap between p and $M^+ = \langle A, C^* \cup C^+, P^* \cup P^+, F^* \cup F^+ \rangle$. We develop searching algorithms in accordance with each chosen resource of open knowledge based on this principle. In each case study we conducted, the robot accumulated knowledge during the process of one task set, but not across task sets, since that would involve a consistency issue regarding the acquired knowledge.

3. Multi-mode NLP

In this section, we demonstrate how we formalize the knowledge in unstructured natural language (e.g., in human-robot dialogue and manual instructions) and in semi-structured natural language (tuples in OMICS) into an intermediate language called Human-Robot Dialogue Structure (HRDS). HRDS captures the semantics of natural language sentences. Its syntax is Lisp-like (see Appendix A), and it can be translated further into the Answer Set Programming (ASP) language (Section 4). HRDS has not been fully developed for handling all situations in human-robot dialogue, but it is sufficient for our needs in this paper. Handling of unstructured and semi-structured knowledge shares the same underlying formalization (i.e., syntactic parsing and semantic interpretation), though the semantic interpretation of the OMICS knowledge needs further processing. By the same mechanism, both Chinese and English are processed with slight differences in configuration, particularly the lexicon. Therefore, our robot can use open knowledge in both languages for one and the same task. However, grammar plays a less important role in the Chinese language, which weakens the performance of the same mechanism in processing Chinese to some extent. This paper presents the multi-mode NLP techniques for English expressions of knowledge.

3.1 Formalizing the knowledge in natural language

The translation process consists of syntactic parsing and semantic interpretation. For the syntactic parsing, the Stanford parser (Klein & Manning, 2003) is employed to obtain the syntax tree of a sentence. The semantic interpretation, using λ-calculus (Blackburn & Bos, 2005), is then applied on the syntax tree to construct the semantics. For this purpose, a lexicon with semantically annotated lexemes and a collection of *augmented syntax rules* (as-rules, for short) are handcrafted in our current implementation. However, Ge & Mooney (2009) provides a promising machine-learning approach to automatic constructing such a lexicon with the as-rules. This approach will be introduced into our work in the future.

Table 1: Part of the lexicon

Word	**Category**	**Semantics**
if	IN	$\lambda p.\lambda q.(\mathbf{cause}\ p\ q)$
the	DT	$\lambda x.\lambda y.y@x$
you	PRP	$\lambda x.x@robot$
will	MD	$\lambda r.r@(t+1)$
press	VBP	$\lambda p.\lambda x.(\mathbf{fluent}\ (\mathbf{pred}\ press\ l\ x\ Y)$ $(\mathbf{conds}\ (\mathbf{pred}\ at\_time\ l\ t)\ p@(\lambda y.y@Y)))$
begin	VB	$\lambda q.\lambda r.\lambda p.(\mathbf{fluent}\ (\mathbf{pred}\ begin\ l\ X\ Y)$ $(\mathbf{conds}\ (\mathbf{pred}\ at\_time\ l\ r)\ p@X\ q@Y))$
microwave	NN	$\lambda x.((\mathbf{pred}\ oven\ x)\ (\mathbf{pred}\ microwave\ x))$
microwave	NN	$\lambda p.\lambda q.(p@q\ (\mathbf{pred}\ microwave\ q))$
oven	NN	$\lambda x.(\mathbf{pred}\ oven\ x)$
START/RESUME	NN	$\lambda p.\lambda q.(p@q\ (\mathbf{pred}\ start\_resume\ q))$
button	NN	$\lambda x.(\mathbf{pred}\ button\ x)$
cooking	NN	$\lambda x.(\mathbf{pred}\ cooking\ x)$
...	...	...

A semantically annotated lexeme lists the syntactic category of a word and its semantics, represented by a λ-calculus formula, as shown in Table 1. One word could have multiple senses (i.e., λ formulae). For example, the word *microwave* has two senses: One represents the concept *microwave oven*, and the other gives the partial semantics of a compound noun (e.g., microwave oven). The combinations of all senses of each word in a sentence will be analyzed in the semantic interpretation.

An as-rule augments the corresponding syntax rule with an extra slot, which combines the semantic interpretations of the parts of the rule's right-hand side. For example, the as-rule VP$(vp := vb@np) \rightarrow$ VB(vb)NP(np) specifies that the semantic interpretation of VP results from applying the semantic interpretation of VB to that of NP. The notation "@" denotes such an application.

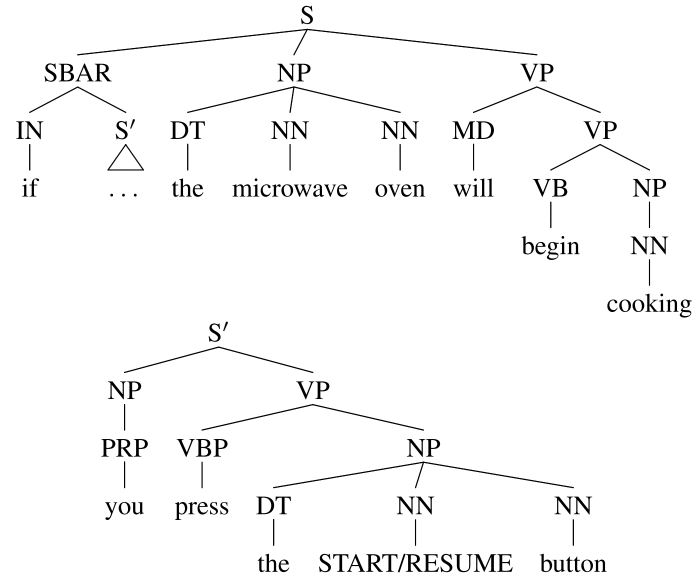

Figure 3. Syntax tree of sentence "*If you press the START/RESUME button, the microwave oven will begin cooking.*"

Some of the as-rules are shown below:

$$
\begin{aligned}
&\text{S}(s := sbar@(np@vp)) \rightarrow \text{SBAR}(sbar)\ \text{NP}(np)\ \text{VP}(vp)\\
&\text{S}(s := np@vp) \rightarrow \text{NP}(np)\ \text{VP}(vp)\\
&\text{SBAR}(sbar := in@s) \rightarrow \text{IN}(in)\ \text{S}(s)\\
&\text{NP}(np := dt@(nn1@nn2)) \rightarrow \text{DT}(dt)\ \text{NN}(nn1)\ \text{NN}(nn2)\\
&\text{NP}(np := nn) \rightarrow \text{NN}(nn)\\
&\text{NP}(np := prp) \rightarrow \text{PRP}(prp)\\
&\text{VP}(vp := md@vp) \rightarrow \text{MD}(md)\ \text{VP}(vp)\\
&\text{VP}(vp := vbp@np) \rightarrow \text{VBP}(vbp)\ \text{NP}(np)\\
&\text{VP}(vp := vb@np) \rightarrow \text{VB}(vb)\ \text{NP}(np)
\end{aligned}
$$

Once the syntax tree of a sentence is generated by the Stanford parser, its semantics are computed using the β-conversion with corresponding lexemes and as-rules. Consider the sentence *If you press the START/RESUME button, the microwave oven will begin cooking*. Its syntax tree is shown in Figure 3. The semantics of words *START/RESUME* of category NN and *button* of category NN, retrieved from the lexicon, are $\lambda p.\lambda q.(p@q\ (\mathbf{pred}\ start\_resume\ q))$ and $\lambda x.(\mathbf{pred}\ button\ x)$, respectively. According to the as-rule NP($np := dt@(nn1@nn2)$) $\rightarrow$ DT(dt) NN($nn1$) NN($nn2$), the phrase *START/RESUME button* is converted to its semantics: $\lambda q.((\mathbf{pred}\ button\ q)\ (\mathbf{pred}\ start\_resume\ q))$. Applying the β-conversion on the syntax tree in Figure 3 from the bottom up to the root, a semantic interpretation of the sentence is gained and

Table 2: A part of the *Tasks/Steps* table

$task$	$stepnum$	$step$
fetch an object	0	locate the object
fetch an object	1	go to the object
fetch an object	2	take the object
fetch an object	3	go back to where you were

expressed as the following HRDS:

$$\begin{aligned}
&(\textbf{cause}\ (\textbf{fluent}\ (\textbf{pred}\ \textit{press}\ l_1\ \textit{robot}\ Y_1)\\
&\qquad\qquad (\textbf{conds}\ (\textbf{pred}\ \textit{at\_time}\ l_1\ t)\ (\textbf{pred}\ \textit{button}\ Y_1)\ (\textbf{pred}\ \textit{start\_resume}\ Y_1)))\\
&\qquad\quad (\textbf{fluent}\ (\textbf{pred}\ \textit{begin}\ l_2\ X_2\ Y_2)\\
&\qquad\qquad (\textbf{conds}\ (\textbf{pred}\ \textit{at\_time}\ l_2\ t+1)\ (\textbf{pred}\ \textit{oven}\ X_2)\ (\textbf{pred}\ \textit{microwave}\ X_2)\\
&\qquad\qquad\quad (\textbf{pred}\ \textit{cooking}\ Y_2))))
\end{aligned}$$

It expresses the causation between two fluents: $(\textbf{pred}\ \textit{press}\ l_1\ \textit{robot}\ Y_1)$ and $(\textbf{pred}\ \textit{begin}\ l_2\ X_2\ Y_2)$. The predicates in the field marked by **conds** of the fluent $\textit{press}$, as well as $\textit{begin}$, are the conditions which the terms (e.g., l_1 and Y_1) in the fluent should satisfy when the fluent is valid. For example, the fluent $(\textbf{pred}\ \textit{press}\ l_1\ \textit{robot}\ Y_1)$ holds under the conditions that Y_1 is a *button* and it functions to *start* or to *resume* the running of a microwave.

3.2 Semantic interpretation of OMICS

In the OMICS project (Gupta & Kochenderfer, 2004), an extensive collection of knowledge was gathered from Internet users in order to enhance the capability of indoor robots for autonomously accomplishing tasks. The knowledge was input into sentence templates by users, censored by administrators, and then converted into and stored as tuples, of which most elements are English phrases. There are 48 tables in OMICS at present, capturing different sorts of knowledge, including a *Help* table (each tuple mapping a user desire to a task that may meet it), a *Tasks* table (containing the names of tasks input by Internet users), and a *Steps* table (each tuple decomposing a task into steps). Since some of the tables are related, some pieces of knowledge in OMICS can be represented as tuples in a joint table generated by an SQL query. The elements of such a tuple are semantically related according to the corresponding sentence templates. Therefore, we introduce *semantically augmented rules*, one for each tuple type, to capture the semantic information of tuples. Some semantically augmented rules are listed below:

$$\begin{aligned}
&\text{Location}(\textit{loc} := (\textbf{state}\ (\textbf{fluent}\ (\textbf{pred}\ \textit{in}\ X\ Y)\ (\textbf{conds}\ \textit{obj}@X\ \textit{room}@Y))))\\
&\qquad \rightarrow \text{Object}(\textit{obj})\ \text{Room}(\textit{room})\\
&\text{TaskSteps}(\textit{tasksteps} := (\textbf{dec}\ \textit{task}\ \textit{step}_0)) \rightarrow \text{Task}(\textit{task})\ \text{Step}(\textit{step}_0)\\
&\text{TaskSteps}(\textit{tasksteps} := (\textbf{dec}\ \textit{task}\ (\textbf{seq}\ \textit{step}_0\ \textit{step}_1))) \rightarrow \text{Task}(\textit{task})\ \text{Step}(\textit{step}_0)\ \text{Step}(\textit{step}_1)\\
&\text{Task}(\textit{task} := \textit{vp}) \rightarrow \text{VP}(\textit{vp})\\
&\text{Step}(\textit{step} := \textit{vp}) \rightarrow \text{VP}(\textit{vp})
\end{aligned}$$

The semantic interpretation of OMICS will be demonstrated with the following example. The *Tasks* table and the *Steps* table combine to produce a joint table *Tasks/Steps*, where each tuple specifies the steps of a task, which also constitutes the definition of the task (Table 2). Accordingly, the

Table 3: The semantic interpretation of tuple elements

Tuple Element	Category	Semantics
fetch an object	Task	$(\mathbf{task}\ fetch\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))$
locate the object	Step	$(\mathbf{task}\ locate\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))$
go to the object	Step	$(\mathbf{task}\ go\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))$
take the object	Step	$(\mathbf{task}\ take\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))$
go back to where you were	Step	$(\mathbf{task}\ go\_back\ X\ (\mathbf{conds}\ (\mathbf{pred}\ location\ X)))$

semantically augmented rule

$$\begin{aligned}&\text{TaskSteps}(tasksteps := (\mathbf{dec}\ task\ (\mathbf{seq}\ step_0\ step_1\ step_2\ step_3)))\\ &\qquad \rightarrow \text{Task}(task)\ \text{Step}(step_0)\ \text{Step}(step_1)\ \text{Step}(step_2)\ \text{Step}(step_3)\end{aligned} \tag{1}$$

defines the semantics of the tuple.

The semantic interpretation procedure works in a bottom-up fashion. The tuple elements are first interpreted as shown in Table 3. Following rule (1), the semantics of tuple elements are combined together piece by piece. Then we get the HRDS representation of the task *fetch an object*:

$$\begin{aligned}&(\mathbf{dec}\ (\mathbf{task}\ fetch\ X_1\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X_1)))\\ &\quad (\mathbf{seq}\ (\mathbf{task}\ locate\ X_2\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X_2)))\\ &\qquad (\mathbf{task}\ go\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))\\ &\qquad (\mathbf{task}\ take\ X\ (\mathbf{conds}\ (\mathbf{pred}\ object\ X)))\\ &\qquad (\mathbf{task}\ go\_back\ X\ (\mathbf{conds}\ (\mathbf{pred}\ location\ X)))))\end{aligned}$$

4. Integrated Decision-making

In the KeJia project, the integrated decision-making module is implemented using Answer Set Programming (ASP), a logic programming language with Prolog-like syntax under stable model semantics originally proposed by Gelfond & Lifschitz (1988). The module implements a growing model $M = \langle A, C^*, P^*, F^* \rangle$, the integrated decision-making mechanism, and some auxiliary mechanisms as an ASP program M^Π. The integrated decision making in M is then reduced to computing answer sets of M^Π through an ASP solver. When the robot's multi-mode NLP module extracts a new piece of knowledge and stores it into M, it will be transformed further into ASP-rules and added into the corresponding part of M^Π.

4.1 Representing growing models in ASP

Given any growing model $M = \langle A, C^*, P^*, F^* \rangle$, the components A, C^*, P^*, and F^* can be represented in ASP with the following conventions. The underlying language includes three pairwise-disjoint symbol sets: a set of *action* names, a set of *fluent* names, and a set of *time* names. The atoms of the language are expressions of the form $occurs(a, t)$ or $true(f, t)$, where a, f, and t are action, fluent, and time name, respectively. Intuitively, $occurs(a, t)$ is true if and only if the action a occurs at time t, and $true(f, t)$ is true if and only if the fluent f holds at time t. Based on these conventions, an ASP-rule is of the form

$$H \leftarrow p_1, \ldots, p_k, not\, q_1, \ldots, not\, q_m.$$

where p_i, $1 \leq i \leq k$, and q_j, $1 \leq j \leq m$ are literals, and H is either empty or a literal. A *literal* is a formula of the form p or $\neg p$, where p is an atom. If H is empty, then this rule is also called a *constraint*. An ASP-rule consisting of only H is called an *ASP-fact*. An ASP program is a finite set of ASP-rules. There are two kinds of negation in ASP, the classical negation $\neg$ and non-classical negation not. Roughly, $not\, q$ in an ASP program means that q is not derivable from the ASP program. Similarly, a constraint that $\leftarrow p_1, \ldots, p_k$ specifies that $p_1, \ldots, p_k$ are not jointly derivable from the ASP program. We take the action *grasp* as an example to show how primitive actions are represented as ASP-rules. The related fluents are

- $grasp(X)$: the action of gripping the object X and picking it up
- $holding(X)$: the fluent that the object X is held in the grip of the robot
- $on(X, Y)$: the fluent that the object X is on the object Y

The effect of executing $grasp(X)$ is $holding(X)$ and described by the following ASP-rules:

$$\begin{aligned} true(holding(X), t+1) &\leftarrow occurs(grasp(X), t). \\ \neg true(on(X,Y), t+1) &\leftarrow occurs(grasp(X), t), true(on(X,Y), t). \end{aligned}$$

Also, the precondition of $grasp(X)$ is $not\, holding(Y)$ for any Y; that is, the grip holds nothing, which is described in ASP as a constraint

$$\leftarrow occurs(grasp(X), t), true(holding(Y), t).$$

Other primitive actions are represented as ASP-rules similarly. In addition, the occurrence of any primitive action is forced to conform to the following restrictions:

$$\begin{aligned} occurs(grasp(X), t) &\leftarrow not\, \neg occurs(grasp(X), t). \\ \neg occurs(grasp(X), t) &\leftarrow not\, occurs(grasp(X), t). \end{aligned}$$

Each element of P^* decomposes a task into sub-tasks or actions. For the general case, see the details in Section 4.2. When a task T is decomposed into an action sequence $\langle a_1, a_2, \ldots, a_n \rangle$, we add an ASP-rule into the ASP program

$$process(T, t, t') \leftarrow occurs(a_1, t), occurs(a_2, t+1), \ldots, occurs(a_n, t+n), t' = t+n.$$

where $process(T, t, t')$ denotes that the task T is accomplished during time t to t'. Accordingly, the definitions of $process(T, t, t')$ are also included in the ASP program. Similarly, for each element of F^* designating a set of literals $\{l_1, \ldots, l_m\}$ to a task T, we add an ASP-rule

$$process(T, t, t') \leftarrow true(l_1, t'), \ldots, true(l_m, t'), t < t'.$$

into the ASP program. This is the case of C^* elements, which are transformed into ASP-rules similarly. Moreover, we use

$$\begin{aligned} &\leftarrow true(holding(X), t), true(falling(X), t). \\ true(falling(X), t) &\leftarrow true(on(X,Y), t), true(falling(Y), t). \end{aligned}$$

to specify that $falling(X)$ is an indirect effect of an action that causes $falling(Y)$ while X is on Y. The *frame problem* is resolved by "inertia laws" of the form

$$\begin{aligned} true(\sigma, t+1) &\leftarrow true(\sigma, t), not\, \neg true(\sigma, t+1). \\ \neg true(\sigma, t+1) &\leftarrow \neg true(\sigma, t), not\, true(\sigma, t+1). \end{aligned}$$

where σ is a meta-variable ranging over fluent names. The inertia laws guarantee the minimal change condition. The initial state (at time 0) of the environment can be expressed in facts of the form $true(\sigma, 0)$.

4.2 Integrated decision-making in ASP

Since any ASP solver innately possesses a general-purpose goal-directed planning schema, we embed a general-purpose task-directed action selection schema into the existing schema, so that the augmentation becomes a general-purpose decision-making mechanism that integrates both schemas and guarantees the executability of every plan it generates when there is sufficient knowledge for the corresponding task. Technically, the augmentation is built on the basis of M^{Π}.

First of all, we name a class of entities called "sequence" as follows: (i) an action a is a sequence; (ii) a task T is a sequence; and (iii) if p_i $(1 \leq i \leq m)$ are sequences, then $p_1; \ldots; p_m$ is a sequence. Let $\tau = \langle s_0, a_0, s_1, \ldots, a_{n-1}, s_n \rangle$ be any trajectory. That τ *satisfies a sequence* p is defined recursively as follows:

(1) If $p = a$, where a is an action, then $a_0 = a$;

(2) If $p = T$, where T is a task such that there is an HRDS rule in P^* that decomposes T into a sequence of sub-tasks, then τ satisfies this sequence of sub-tasks, or where T is a task such that it is designated as a set of literals in F^*, then this set is a subset of the state s_n;

(3) If $p = p_1; \ldots; p_m$, where p_i $(1 \leq i \leq m)$ are sequences, then there exist $0 \leq n^1 \leq n^2 \leq \cdots \leq n^{m-1} \leq n$ such that:

– the trajectory $\langle s_0, a_0, \ldots, s_{n^1} \rangle$ satisfies p_1;

– the trajectory $\langle s_{n^1}, a_{n^1}, \ldots, s_{n^2} \rangle$ satisfies p_2;

– $\cdots$;

– the trajectory $\langle s_{n^{m-1}}, a_{n^{m-1}}, \ldots, s_n \rangle$ satisfies p_n.

According to the definitions above, if a trajectory $\langle s_0, a_0, s_1, \ldots, a_{n-1}, s_n \rangle$ satisfies a sequence $a; a'$ where a and a' are actions, and a' is not executable in s_1, then $a_0 = a$ and there exists a state s_m $(1 \leq m \leq n)$ such that s_m satisfies the preconditions of a' and $a_m = a'$. In other words, the sub-trajectory $\langle s_1, a_1, \ldots, s_m \rangle$ fills the "gap" between a and a'.

A sequence S specifies how to complete a task T step by step. If a trajectory contains a sub-trajectory which satisfies S, then the corresponding task T is also completed in this trajectory. Now we consider how to specify a sequence S in ASP. Given an growing model M, we want to obtain a set of ASP-rules of S, Π_S, such that a trajectory $\langle s_0, a_0, s_1, \ldots, a_{n-1}, s_n \rangle$ satisfies both M and S if and only if $\{true(\sigma, i) | \sigma \in s_i, 0 \leq i \leq n\} \cup \{\neg true(\sigma, i) | \neg\sigma \in s_i, 0 \leq i \leq n\} \cup \{occurs(a_i, i) | 0 \leq i \leq n-1\}$ is an answer set of $M^{\Pi} \cup \Pi_S$.

Any sequence defined above can be specified in HRDS as $\langle sequence \rangle$; see Appendix A for details. Given such a sequence S, we define the set Π_S of ASP-rules recursively as follows (where t, t', t_1, $\ldots$, t_{m-1} are meta-variables ranging over time):

(1) If $S \; = \; (\mathbf{act}\; name\_a\; X_1 \ldots X_m$
$\quad (\mathbf{conds}\; (\mathbf{pred}\; cond_1\; X_1 \ldots X_m) \ldots (\mathbf{pred}\; cond_n\; X_1 \ldots X_m)))$
where $name\_a$ is an action name, $X_1 \ldots X_m$ are its parameters, and predicates $(\mathbf{pred}\; cond_1\; X_1 \ldots X_m), \ldots, (\mathbf{pred}\; cond_n\; X_1 \ldots X_m)$ represent domains of these parameters, then Π_S is the set of the following ASP-rules

$$complete(p, t, t+1) \leftarrow occurs(name\_a(X_1, \ldots, X_m), t),$$
$$true(cond_1(X_1, \ldots, X_m), t), \ldots, true(cond_n(X_1, \ldots, X_m), t).$$

(2) If $S \; = \; (\mathbf{task}\; name\_t\; X_1 \ldots X_m$
$\quad (\mathbf{conds}\; (\mathbf{pred}\; cond_1\; X_1 \ldots X_m) \ldots (\mathbf{pred}\; cond_n\; X_1 \ldots X_m)))$
where $name\_t$ is a task name, $X_1 \ldots X_m$ are its parameters, and predicates $(\mathbf{pred}\; cond_1\; X_1 \ldots X_m), \ldots, (\mathbf{pred}\; cond_n\; X_1 \ldots X_m)$ represent domains of these param-

eters, then Π_S contains

$$complete(p,t,t') \leftarrow complete(name\_t(X_1,\ldots,X_m),t,t'),\\ true(cond_1(X_1,\ldots,X_m),t),\ldots,true(cond_n(X_1,\ldots,X_m),t).$$

(3) If P^* designates a sequence S' to accomplish the task $name\_t(X_1,\ldots,X_m)$, then Π_S also contains $\Pi_{S'}$ and

$$complete(name\_t(X_1,\ldots,X_m),t,t') \leftarrow complete(S',t,t').$$

(4) If F^* designates a set of literals $\{\sigma_1,\ldots,\sigma_o,\neg\sigma_{o+1},\ldots,\neg\sigma_m\}$ for the task, then Π_S also contains

$$complete(name\_t(X_1,\ldots,X_m),t,t') \leftarrow true(\sigma_1,t'),\ldots,true(\sigma_o,t'),\\ \neg true(\sigma_{o+1},t'),\ldots,\neg true(\sigma_m,t'),\ t<t'.$$

(5) If $S = (\textbf{seq}\ S_1 \ldots S_m)$, where S_i $(1 \le i \le m)$ are sequences, Π_S contains $\Pi_{S_1},\ldots,\Pi_{S_m}$ and

$$process(S;t;t') \leftarrow process(S_1,t,t_1), process(S_2,t_1,t_2),\ldots,process(S_m,t_{m-1},t').$$

5. Case studies

We have conducted a variety of case studies of open knowledge enabling on KeJia robots. In one case study, we tested the KeJia robot's ability to acquire causality knowledge through human-robot dialogue (Chen et al., 2010). A testing instance is shown in Figure 4. A board was set on the edge of a table, with one end sticking out. A red can was put on the end of the board that stuck out, and a green can was placed on the other end. The task for KeJia was to pick up the green can under a default presupposition of avoiding anything falling. We took a version of KeJia without any built-in knowledge about "balance", "fall", or other equivalents. A human told the robot, "*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.*" KeJia's NLP module extracted the knowledge and transformed it into ASP-rules. Using these the rules, the decision-making module generated a plan, in which the robot moved the red can to the table first and then picked up the green one. The robot accomplished the task by executing this plan.[2] It is worthwhile emphasizing the fact that KeJia could not have accomplished the task without the causality knowledge acquired. This indicates that KeJia's ability is substantially enhanced by knowledge acquisition through human-robot dialogue. In another case study (Xie, Chen, Ji, & Sui, in press), a user asked the robot to heat up some popcorn in a microwave oven, while the robot had not known the functions of the buttons on the control panel before the experiment. The robot extracted knowledge of the buttons' function from the manual in English (Section 3.1 shows the translation of one of the sentences from the manual). With the extracted knowledge, the robot generated a plan consisting of 22 primitive actions in 3.4 seconds. It executed the plan and accomplished the task in 11.3 minutes.[3]

In this paper, we report a new evaluation of the proposed techniques through two tests on two large task sets collected from OMICS. We took a version of the OK-KeJia robot that contains only two sorts of built-in knowledge. One is that of action models, and the other is the semantically annotated lexemes of words in the lexicon, a type of linguistic knowledge (see Section 3.1) which is only used in the multi-mode NLP module. The open knowledge the robot could gather for the task planning in the experiments was limited to two tables of OMICS and the synonymies of *WordNet*,[4]

[2] Demo video available at http://ai.ustc.edu.cn/en/demo/Cause_Teach.php

[3] Demo video available at http://ai.ustc.edu.cn/en/demo/ServiceRobot_oven.php

[4] http://wordnet.princeton.edu/wordnet/

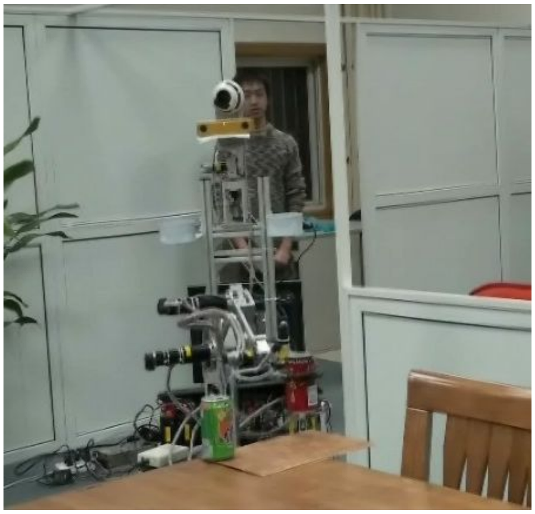

Figure 4. A test instance of acquiring causality knowledge through human-robot dialogue

without any handcrafted knowledge.

Each test varied in two dimensions, the action model and the open knowledge base. Five action models, $AM_1 = \{move\}$, $AM_2 = \{move, find\}$, $AM_3 = \{move, find, pick\_up\}$, $AM_4 = \{move, find, pick\_up, put\_down\}$ and $AM_5 = \{move, find, pick\_up, put\_down, open, close\}$, were chosen in order to examine the impact of the different action capabilities of a robot on its overall performance. Each test consisted of three rounds with different open knowledge bases, in order to show the impact of open knowledge on performance.

The task set in Test 1 was defined as follows: There are 11,615 different tuples in *Tasks/Steps*, each consisting of a task name T and a sequence of steps (sub-tasks), s. We take s as the definition of T. For example, there are two tuples: <*help someone carry something, 0. pick up the item, 1. walk with the item to where the person needs it*> and <*help someone carry something, 0. get object, 1. follow person*>. Obviously, these two tuples with the same task name define two different tasks, because there is no guarantee that any action sequence that fulfills one of the two tasks must fulfill the other one. Therefore, we collected all 11,615 task definitions in the task set for Test 1.

In the first round of Test 1, only the action models were used in the planning procedure, with no open knowledge (i.e., $C^* = P^* = F^* = \emptyset$). Every task in the task set was input into the robot and the robot tried to fulfill it. A task was solvable by the robot's planner if and only if each step of the task was a primitive action. In the second round, 11,615 *Tasks/Steps* tuples were used as a sort of procedural knowledge, now taken as task-decomposition rules (td-rules, for short). Hence $C^* = F^* = \emptyset$ and $P^* =$ *Tasks/Steps*. It follows that a task could be fulfilled if all of its steps could be reduced to primitive actions through td-rules. In the third round, the robot tried to get open knowledge from both *Tasks/Steps* and WordNet (i.e., $P^* =$ *Tasks/Steps*, $C^* \subseteq$ WordNet synonymies, and $F^* = \emptyset$). Consequently, "*move to a location*" and "*go to a location*" were identified as equivalent and executable by the robot, although only the former could be matched to the robot's primitive action. Obviously, the open knowledge used in this test was extremely sparse—only the definitions of the tasks were used as the procedural knowledge.

We developed a set of algorithms in the case study. Algorithm 1 is only for understanding the main ideas of the actual algorithm for the second-round experiment of Test 1. In this iteration, a

Algorithm 1 $getActionSequence$(phrase ph)

```
/* generate an action sequence for task $ph$ */
$p := parsePredicates(ph)$
/* semantically parses $ph$ to internal representation $p$ */
$Subtask := subTask(p)$
/* initiate $Subtask$ with sub-tasks of $p$ */
$P^+ := P^*$
/* initiate $P^+$ with the action model $P^*$ */
$Res := taskPlan(p)$
/* $taskPlan$ returns an action sequence computed by the integrated decision-making module */
if $Res \neq null$ then
    return $Res$
end if
while there is a new tuple $t$ from Tasks/Steps that matches an element of $Subtask$ do
    $q := parsePredicates(t)$
    $Subtask := subTask(q)$
    $P^+ := P^+ \cup q$
    $Res := taskPlan(p)$
    if $Res \neq null$ then
        return $Res$
    end if
end while
return $Failure$
```

new td-rule from the *Tasks/Steps* table was selected, processed by the multi-mode NLP module, and added into the growing model. Then the planner was called on to compute an action sequence for the user task. For the third-round, synonymies of WordNet were used to substitute concepts that appeared in td-rules with their equivalent primitive actions. The planner $taskPlan$ in the algorithm is a simplified version of the integrated decision-making module due to some implementation issues. Most importantly, since this case study was conducted on large task sets collected from OMICS, we did not succeed in associating with each task an observation, which were obtained by our real robots in other case studies mentioned above. Consequently, the executability condition was simplified to "consisting of primitive actions".

The task set in Test 2 consisted of 467 different desires that appeared in the *Help* table of OMICS, with duplicate ones discarded. Some examples of *Help* tuples are <*are sick, by giving medicine*>, <*are cold, by making hot tea*>, and <*feel thirsty, by offering drink*>. In each tuple, the first element is taken as a user desire, while the second element, a task, is taken as a means to meet the desire. Some of these tasks may appear in *Tasks/Steps* and thus the *Tasks/Steps* table provides a resource of open knowledge to meet desires in *Help*. Because of its nature, a single desire d appeared in a tuple $<d, t>$from *Help* can be met by any t appeared in *Tasks/Steps*. Therefore, the open knowledge used in Test 2 was much less sparse than that in Test 1. In the first round of Test 2, no open knowledge was used as in Test 1. In the second round, all 3,405 unduplicated tuples in *Help* were taken as functional knowledge and all in *Tasks/Steps* as procedural knowledge (i.e., $F^* =$ *Help*, $P^* =$ *Tasks/Steps*, and $C^* = \emptyset$). WordNet synonymies were added as conceptual knowledge in the third round.

The experimental results are shown in Table 4. On every action model in each round, the number of tasks or desires that were fulfilled by the robot is listed in the table. In addition, the percentages of fulfilled tasks or desires with respect to the size of the task sets on AM_5 are listed in the last column.

Table 4: Experimental results

Open Knowledge	$\mathbf{AM_1}$	$\mathbf{AM_2}$	$\mathbf{AM_3}$	$\mathbf{AM_4}$	$\mathbf{AM_5}$	**Percentage on** $\mathbf{AM_5}$
Test 1 (11,615 user tasks)						
Null	6	24	45	164	207	1.78%
Tasks/Steps (11,615 rules)	7	28	51	174	219	1.89%
Tasks/Steps+WordNet	16	43	71	233	297	2.56%
Test 2 (467 user desires)						
Null	0	1	1	4	4	0.86%
Help+Tasks/Steps (15,020 rules)	29	63	83	107	117	25.05%
Help+Tasks/Steps+WordNet	43	73	87	119	134	28.69%

We make the following observations.

(1) **The overall performance increased remarkably due to the use of a moderate amount of open knowledge**. The percentage of fulfilled tasks increased from 1.78% to 2.56% in Test 1 with very sparse open knowledge of two types, and from 0.86% to 28.69% in Test 2 with a moderate amount of open knowledge of three types, respectively. The difference in the performance improvements in the two tests further reveals the significant function of the amount or "density" of the open knowledge.

(2) **General-purpose techniques played an essential role in supporting the use of open knowledge**. It is worthwhile emphasizing that the improvements were made through using open knowledge via general-purpose mechanisms (multi-mode NLP, integrated decision-making and open knowledge searching), without any manual assistance. Actually, it is difficult to utilize manual assistance when task sets are large.

(3) **A robot's basic ability (primitive actions) was still a key factor for the overall performance**. As expected, the robot met more user requests with more primitive actions in all the cases. More detailed examination indicates that there were many user requests that were not met due to the powerlessness of the action models used in the experiments, though KeJia robots can do more actions that were not contained in the action models.

6. Conclusions

As part of the remarkable progress happening in HRI-related areas, it is interesting to consider the possibility of improving the performance of robots autonomously using open knowledge instead of handcraft-coded knowledge. The KeJia project is a long-term effort toward this goal. We focused on three issues in this paper: (i) how a robot understands and extracts open knowledge in unstructured and semi-structured natural language; (ii) how a robot makes use of different types of knowledge in decision making in order to meet user requests; (iii) how a robot can be aware of what knowledge it lacks for a given task and search for the missing knowledge from open-source resources. Using techniques for multi-mode NLP, integrated decision making, and open knowledge searching, we demonstrated the overall performance of an OK-KeJia robot increases remarkably with the use of appropriate open knowledge. Considering that only a very small proportion of knowledge in OMICS (i.e., 2 tables out of 48) was used in this case study, there would be room for further progress.

Many challenges remain. The tests with large task sets conducted so far did not involve the context of HRI, in which a user request can sometimes only be understood together with an observation of the environment. This gap suggests an investigation into the connection between open knowledge and HRI contexts. How to make use of a larger amount of open knowledge is also an interesting issue, especially when context is considered. We believe these two issues are related. They may demand more research on techniques for multi-mode NLP, integrated decision making, and identification of knowledge gaps. In addition, a more powerful NLP module is needed and would further improve the robot's performance. For example, there are now about 7.6% and 50.3% of tuples from the *Steps* table and the *Help* table, respectively, that cannot be handled by the current NLP module. Another question that remains is about the relation between open knowledge and action model learning. Integration of both techniques may help further improve the performance of human-robot interactions.

Acknowledgments

There are about 30 Chinese professors and graduate students working on the KeJia Project, which has been supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grants 60745002 and 61175057, as well as the USTC Key Direction Project and the USTC 985 Project. The authors are grateful to the other team members, the sponsors, and the following persons for helpful discussions about research issues in the project: Michael Gelfond, David Hsu, Fengzhen Lin, Daniele Nardi, Peter Stone, Manuela Veloso, Mary-Anne Williams, Yan Zhang, and Shlomo Zilberstein. We also thank the anonymous reviewers for their valuable comments and suggestions on the manuscript of this work.

Appendix A

The BNF definition for the syntax of Human-Robot Dialogue Structure

$\langle \mathit{variable} \rangle$ ::= a string where the first character is upper-case, e.g., X

$\langle \mathit{constant} \rangle$ ::= a string where the first character is lower-case or a underline, e.g., a, $\_b$, etc.

$\langle \mathit{term} \rangle ::= \langle \mathit{constant} \rangle \mid \langle \mathit{variable} \rangle$

$\langle \mathit{predicate} \rangle ::= (\mathbf{pred}\ \langle \mathit{name} \rangle\ \langle \mathit{term} \rangle^{+})$

$\langle fluent \rangle ::= (\mathbf{fluent}\ \langle \mathit{predicate} \rangle\ (\mathbf{conds}\ \langle \mathit{predicate} \rangle^{+}))$

$\langle formula \rangle ::= \langle fluent \rangle \mid (\mathbf{neg}\ \langle formula \rangle) \mid (\mathbf{conj}\ \langle formula \rangle^{+}) \mid (\mathbf{disj}\ \langle formula \rangle^{+})$

$\langle \mathit{action} \rangle ::= (\mathbf{act}\ \langle \mathit{action\_name} \rangle\ \langle \mathit{term} \rangle^{+}\ (\mathbf{conds}\ \langle \mathit{predicate} \rangle^{+}))$

$\langle \mathit{task} \rangle ::= (\mathbf{task}\ \langle \mathit{task\_name} \rangle\ \langle \mathit{term} \rangle^{+}\ (\mathbf{conds}\ \langle \mathit{predicate} \rangle^{+}))$

$\langle \mathit{sequence} \rangle ::= \langle \mathit{action} \rangle \mid \langle \mathit{task} \rangle \mid (\mathbf{seq}\ \langle \mathit{sequence} \rangle^{+})$

$\langle \mathit{causation} \rangle ::= (\mathbf{cause}\ \langle fluent \rangle\ \langle fluent \rangle)$

$\langle \mathit{decomposition} \rangle ::= (\mathbf{dec}\ \langle \mathit{task} \rangle\ \langle \mathit{sequence} \rangle)$

$\langle \mathit{effect} \rangle ::= (\mathbf{eff}\ \langle \mathit{task} \rangle\ \langle formula \rangle)$

$\langle \mathit{statement} \rangle ::= (\mathbf{state}\ \langle formula \rangle)$

$\langle \mathit{request} \rangle ::= (\mathbf{req}\ \langle \mathit{task} \rangle)$

References

Blackburn, P., & Bos, J. (2005). *Representation and inference for natural language: A first course in computational semantics*. Chicago, USA: CSLI Publications.

Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., et al. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, *114*(1-2), 3–55 http://dx.doi.org/10.1016/S0004-3702(99)00070-3.

Cantrell, R., Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S., & Scheutz, M. (2012). Tell me when and why to do it!: Run-time planner model updates via natural language instruction. In *Proceedings of the 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI-12)* (pp. 471–478 http://dx.doi.org/10.1145/2157689.2157840). Boston, USA: ACM.

Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F., & Xie, J. (2010). Developing high-level cognitive functions for service robots. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)* (pp. 989–996). Toronto, Canada: IFAAMAS.

Chen, X., Sui, Z., & Ji, J. (2012). Towards metareasoning for human-robot interaction. In *Proceedings of the 12th International Conference on Intelligent Autonomous System (IAS-12)* (p. 355-367 http://dx.doi.org/10.1007/978-3-642-33932-5_34). Jeju Island, Korea: Springer Berlin Heidelberg.

Doshi, F., & Roy, N. (2007). Efficient model learning for dialog management. In *Proceedings of the 2nd ACM/IEEE International Conference on Human Robot Interaction (HRI-07)* (pp. 65–72 http://dx.doi.org/10.1145/1228716.1228726). Arlington, Virginia, USA: ACM.

Fong, T., Thorpe, C., & Baur, C. (2003). Robot, asker of questions. *Robotics and Autonomous Systems*, *42*(3), 235–243 http://dx.doi.org/10.1016/S0921-8890(02)00378-0.

Ge, R., & Mooney, R. (2009). Learning a compositional semantic parser using an existing syntactic parser. 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 (ACL-IJCNLP-09)* (pp. 611–619 http://dx.doi.org/10.3115/1690219.1690232). Singapore: ACL.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming (ICLP-88)* (pp. 1070–1080). Seattle, Washington: MIT Press.

Gupta, R., & Kochenderfer, M. (2004). Common sense data acquisition for indoor mobile robots. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)* (pp. 605–610). San Jose, California, USA: AAAI Press / MIT Press.

Kaupp, T., Makarenko, A., & Durrant-Whyte, H. (2010). Human-robot communication for collaborative decision making–a probabilistic approach. *Robotics and Autonomous Systems*, *58*(5), 444–456 http://dx.doi.org/10.1016/j.robot.2010.02.003.

Klein, D., & Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL-03)* (pp. 423–430 http://dx.doi.org/10.3115/1075096.1075150). Sapporo Convention Center, Sapporo, Japan: ACL.

Kruijff, G., Lison, P., Benjamin, T., Jacobsson, H., Zender, H., Kruijff-Korbayová, I., et al. (2010). Situated dialogue processing for human-robot interaction. *Cognitive Systems*, *8*, 311–364 http://dx.doi.org/10.1007/978-3-642-11694-0_8.

Lemaignan, S., Ros, R., Sisbot, E., Alami, R., & Beetz, M. (2012). Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction. *International Journal of Social Robotics*, *4*(2), 181–199 http://dx.doi.org/10.1007/s12369-011-0123-x.

Levesque, H., Reiter, R., Lesperance, Y., Lin, F., & Scherl, R. (1997). GOLOG: A logic programming language for dynamic domains. *The Journal of Logic Programming*, *31*(1-3), 59–83 http://dx.doi.org/10.1016/S0743-1066(96)00121-5.

Rosenthal, S., Biswas, J., & Veloso, M. (2010). An effective personal mobile robot agent through symbiotic human-robot interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)* (pp. 915–922). Toronto, Canada: IFAAMAS.

Rosenthal, S., Veloso, M., & Dey, A. (2011). Learning accuracy and availability of humans who help mobile robots. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11)* (pp. 60–74). San Francisco, California, USA: AAAI Press.

Talamadupula, K., Benton, J., Kambhampati, S., Schermerhorn, P., & Scheutz, M.(2010). Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *1*(2), 14:1–14:24 http://dx.doi.org/10.1145/1869397.1869403.

Tenorth, M., & Beetz, M. (2009). KnowRob—-knowledge processing for autonomous personal robots. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-09)* (pp. 4261–4266 http://dx.doi.org/10.1109/IROS.2009.5354602). St. Louis, MO, USA: IEEE.

Thrun, S. (2004). Toward a framework for human-robot interaction. *Human–Computer Interaction*, *19*(1-2), 9–24 ttp://dx.doi.org/10.1207/s15327051hci1901&2_2.

Xie, J., Chen, X., Ji, J., & Sui, Z. (in press). Multi-mode natural language processing for extracting open knowledge. In *Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology (IAT-12).*

Authors names and contact information: X.-P. Chen, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. Email: xpchen@ustc.edu.cn; J.-K. Xie, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. Email: devilxjk@mail.ustc.edu.cn; J.-M. Ji, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. Email: jianmin@ustc.edu.cn; Z.-Q. Sui, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. Email: zqsui@mail.ustc.edu.cn