Robotics and Autonomous Systems 140 (2021) 103763

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Ontology-based knowledge management with verbal interaction for command interpretation and execution by home service robots

L. Villamar Gómez *, J. Miura

Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Aichi, 441-8580, Japan

ARTICLE INFO

Article history:
Received 17 February 2020
Received in revised form 26 February 2021
Accepted 1 March 2021
Available online 4 March 2021

Keywords:
Knowledge management
Ontology-based
Service robots
Human–robot interaction

ABSTRACT

This paper describes a system for service robots that combines ontological knowledge reasoning and human–robot interaction to interpret natural language commands and successfully perform household chores, such as finding and delivering objects. Knowledge and context reasoning is essential for providing more efficient service robots, given their diverse and continuously changing environments. Moreover, since they are in contact with humans, robots require such skills as interaction and language. Therefore, we developed a system with specific modules to manage robots' knowledge and reasoning, command analysis, decision-making, and talking interaction. The system relies on inference methods and verbal interaction to understand commands and clarify uncertain information. We tested our system inside a simulated environment where the robot receives commands with missing or unclear information. The system's performance was compared with the average performance of human subjects who completed the same commands in the simulation.

1. Introduction

Robots are becoming a part of our daily lives in different aspects; they now work as hotel receptionists or waitresses in restaurants, among others. Service robotics, which refers to assistant robots at home, has been gaining importance. Researchers are creating robots to accompany and fulfill the needs of elderly or disabled humans.

Similar to any other machine, robots should be able to execute specific tasks successfully. However, service robots need other skills and characteristics to be involved in the social environment of humans. Several factors must be considered for service robots, such as how they can interact, their response and speed, and their usefulness for their tasks. A robot's behavior is crucial during human–robot contact; the person should feel safe, willing to be assisted, and rely on the robot, and the overall experience must be satisfactory [1]. An essential element needed to provide efficient service robots is knowledge and context reasoning, given their diverse and continuously changing environments.

Various researchers have studied how to provide the knowledge that robots need to complete different tasks [2–5]. Some have attempted to make robots learn new concepts or assignments by themselves [6–8]. Reasoning based on acquired knowledge before taking action is a skill be pursued in robots [9–11]. Most of these published works focus on specific and separate skills that service robots need. However, some of them lack well-ordered concepts about the environment, and the knowledge is limited to that acquired during the learning phase that is, the first human–robot interactions. In other cases, the questions' and answers' patterns are fixed and do not handle unexpected answers from users [3].

The implementation of natural language in robots has been explored at different levels of human–robot interaction. Models that can extract information from natural language instructions and their surrounding environments have been developed to improve robots' instruction understanding. NL-based probabilistic, cognitive, and logic models are used for plan generation. In contrast, theoretical knowledge grounding, knowledge gap detection, and gap-filling models have been developed for knowledge world mapping [12].

In this paper, we develop a system for service robots that combines ontological knowledge reasoning and human–robot interaction to interpret natural language commands and successfully perform household chores, such as finding and delivering objects. We use an ontology to represent the general information of the components in the environment and their relationships; moreover, the system links natural language commands, the ontology object representation, and the information of the real objects involved. The robot disambiguates uncertain requests through spoken interaction with the human before completing a task. It utilizes information from the ontological knowledge to create more precise questions.

* Corresponding author.
E-mail addresses: liliana@aisl.cs.tut.ac.jp (L. Villamar Gómez), jun.miura@tut.jp (J. Miura).

https://doi.org/10.1016/j.robot.2021.103763
0921-8890/

The contributions of this work are the (a) conjunction of knowledge reasoning and verbal interaction to interpret and disambiguate natural language commands, (b) knowledge management based on ontology with inference capabilities in a home environment, and (c) question formulation incorporating ontological assertions. Moreover, the system's functionality and performance are demonstrated by experiments in a simulated environment and through comparison with human behavior.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 shows an overview of the entire system and briefly explains its central modules and their functions. Section 4 explains how knowledge is organized and accessed. Section 5 describes the process the robot conducts to understand a command before executing it. Section 6 shows the experiments performed using a simulated environment and explains the results. The paper ends with the conclusions and discussion in Section 7.

2. Related work

2.1. Ontology and knowledge-based systems

The first step in creating a service robot with the abovementioned characteristics is to provide it with knowledge not only for completing chores but also for understanding what is being asked, to manage unexpected situations or ambiguous assignments.

Numerous systems have been developed to manage knowledge in different areas. For instance, in [13], the authors emphasized the importance of managing knowledge for software maintenance that is hard to track using conventional documentation methods. They proposed the use of ontologies to save such knowledge and described a methodology for developing such an ontology.

A model conception for enterprise knowledge management was proposed in [14]. The model includes a set of ontologies specializing in specific parts of the process of knowledge management, such as the knowledge acquisition, knowledge storage, and knowledge identification processes. Another system proposing a merged ontology was presented in [15], which includes medical, hotel, and city ontologies.

An application for knowledge management in mechatronics was developed in [16]; it contains retrieval information methods that humans and computers can understand. However, the knowledge is static and does not consider the dynamic management of the information, which is essential in a changing environment. The authors of [3] described an attempt to create cognitive maps through interactive dialog; new attributes and names of objects can be learned through the interaction considering the user preference toward an object. Nevertheless, the acquired knowledge is limited to the user's response and does not consider solving inconsistencies in knowledge.

In the area of robotics, different studies have proposed the use of ontologies as a standardization of knowledge representation. Ontologies have been proposed to describe robots, parts of robots, and their relationships [17]; products and their final assembled states [18]; and appliances, their moving parts, and their functionalities [19].

Some ontologies were designed to include concepts related to advanced driver-assistance systems, driving tasks, and driving distractions [20]. To eliminate the heterogeneity between devices and applications, some ontology models describe concepts on the basis of sensor-stimulus-observation design patterns [4].

However, the knowledge described in these ontologies needs to be complete, including all the parts involved and their functions, because inferences rely on the integrity and precision of the represented knowledge. Moreover, these approaches do not consider any mechanism besides the inference process for dealing with lack of information.

A more robust approach is KnowRob [21], a knowledge processing system designed to give entirely autonomous robots knowledge to accomplish manipulation tasks. This system includes knowledge representation, reasoning techniques, and methods of knowledge acquisition and exchange. Its applicability has been demonstrated through experiments of a robot making pancakes [22] and in projects such as openEASE [23].

Two important aspects are the use of WordNet to disambiguate vague natural language task descriptions [24] and the integration of robot control data into symbolic reasoning to generate information needed to execute tasks. However, the symbolic representation used in the study was relatively shallow and customized to produce functional knowledge to execute tasks, and the consistency in the knowledge base was not considered. Hence, executing symbolic inference processes with a large amount of implicitly encoded information in the control system might increase the computational cost. Moreover, the disambiguation process focused on creating a detailed robot plan of actions to be executed; by contrast, our objective is to disambiguate the objects required in the actions and complete the plan to be executed.

Recently, KnowRob was extended and partially redesigned, and the new release was introduced as KnowRob 2.0. It integrates photorealistic rendering and acquisition of low-level robot data and information concerning tasks, contexts, and goals, among others [25]. KnowRob 2.0 implements hybrid reasoning, unifying inner world knowledge, virtual and logical knowledge, and knowledge acquired from perceptual data during task execution. It also includes a question-answering feature enabled by Prolog in the interface shell.

However, potential queries are meant to request manipulation information or motion parameters needed for a task. It does not consider actual human interaction to cooperate or support task decisions in dynamic plan generation. Furthermore, as stated in [25], the knowledge originated from data collected from different sources might be redundant and inconsistent, leading to the computation of multiple hypotheses to determine the correct answer. Moreover, the paper did not mention how to deal with contradictory information.

Our purpose is to reason on the natural language command against the knowledge base to identify vague commands in terms of the objects needed for a task rather than the motions. Moreover, we include information derived from human interaction to support task completion when doubts arise about the involved entities and the reasoning cannot resolve them.

2.2. Human–robot interactive systems

Apart from knowledge, service robots need the ability to interact with humans. Tutoring systems are an example of human–robot interaction. The use of robots in one-on-one tutoring promotes learning gains and strengthens engagement [26]. The combination of the use of a robot's gestures and adaptive training leads to better learning outcomes in children [27].

Systems relying on learning-by-teaching methods using robots that act as learners and receive correction from children increase children's abilities and overall performance [6]. Methods of finding the best strategies for teaching robots have been studied in several works. In particular, physical interactions and reductions in unintended learning enhance robot's learning efficiency [7].

Means of interaction is a key factor in human–robot interactive systems. In [8], a human–robot architecture for interactive learning and conceptual reasoning using ontologies was

described. The interaction is based on natural language communication, and it can learn concepts such as actions and targets of actions, although low-level mechanisms for linking the concepts to physical actions were not considered.

Teaching and learning strategies are also prominent in human–robot interaction. Correct strategies or combinations of strategies can be used to interpret the meaning of human actions well, since every user interacts differently [9]. Other tools can be used to reduce the physical and mental demands of humans when they collaborate with robots. These include an interactive combination of projection and touch-enabled table and kinesthetic teaching with a high level of abstraction [28]. However, users might take some time to learn the correct method of using such tools.

2.3. Dialog systems

The usage of spoken language to interact with systems is a natural way of communication in humans. Many works have attempted to use natural language as the primary means of communication. In [29], scene generation was achieved by using natural language, and it can be dramatically improved by online image cloud retrieval, and synthesized scene refinement through natural language.

Some widespread usages of dialog systems are chatbots, recommendation systems, and feedback collection. Sufficient and convincing information improves user satisfaction with virtual dialog [30]. Letting on the user's initiative creates a more appealing and entertaining experience [31], and gathering guest feedback with robots is a convenient and useful method for the hospitality industry [32]. Dialog systems also emphasize adaptation to new domains and changes in the knowledge content with new information [33].

Conversational robots are a central topic regarding the future of human–robot communication. Multiagent conversations and interruptions must be managed appropriately while considering the importance of different factors, such as the conversation topics and emotional behaviors [10]. Moreover, a user's experience in establishing dialogs with robots can be enhanced by switching comment-speaking between more than one robot and the human to increase the impression of dialog continuity and avoid breakdowns [34]. The robot can increase the engagement, attention, and understanding of the user by adapting its dialog through considering verbal and nonverbal feedback [35].

The use of dialog-based interaction is an important skill for robots socializing with humans. However, rather than creating dialogs for entertainment purposes, a home service robot needs to create context and environment-specific conversations to solve requests accurately. The dialog creation needs to incorporate grounded knowledge, including general and situation-dependent concepts. The information received through the conversation is essential for task planning and execution.

2.4. Service robots

Systems and service robots that are in contact with humans require several features, such as general knowledge, interaction skills, and language skills. When a robot deals with a verbal request, it sometimes needs to clarify the elements involved in the human request. Research has been performed to evaluate different visualization mechanisms that will allow a user to determine objects that match a request, and to develop more features to help robots select correct ones, such as head-mounted displays, monitors, and projectors [36].

A robot requires techniques to connect verbs to their associated tasks intuitively and the tools or sensors needed to execute such tasks [37]. A study expanded a robot's language understanding and grounded physical objects through conversations with humans [38]. The system creates dialogs to confirm the understood command or clarify unknown words to later ground the concept. It incorporates the description of objects, including visual, audio, and haptic properties. Although the system can learn new concepts referring to object properties successfully, it still requires a considerable number of clarification questions, even after training perception and parsing modules, to understand before executing a command.

Language understanding is essential for service robots. Some probabilistic graphical models are trained by associating language to scene semantics and perceptual classifiers to map natural language instructions correctly [11]. However, these models do not consider techniques of solving ambiguous instructions, such as verbal interaction.

The authors of [39] proposed an approach that combines existing inference technologies to identify a command's semantic information and improve the design of human–robot interaction architectures. This work was extended in [40] with the incorporation of discriminative learning and distributional semantics. Spoken language understanding is achieved using linguistic data and perceptual knowledge. The resulting semantic frames are mapped into plans whose respective arguments are associated with their corresponding actors. Although this approach deals with ambiguous sentence structures, it does not consider situations where knowledge is missing in the semantic map, and neither creates any spoken interaction other than upon receiving the initial command.

Another approach to language understanding was developed in [41]; in this approach, the interpretation of utterances and context information are represented as logical forms. Then, given a first-order formula, the framework uses theorem provers along with model builders as inference engines to find either a proof or a model that satisfies it. Despite the robustness of this approach, the search for a suitable model or proof might require several processes; thus, it will need a long time to solve a task but still not guarantee that the identified model is the most efficient. Although this approach implements clarification dialog when the recognition confidence is low, it does not consider the number of dialogs made before being able to obtain the information needed for the model or the proof.

In [42], a log-linear reward function model was presented to find the possible sequence of instructions considering environment context information to perform tasks, such as cooking (ramen). This method handles generalization to new environments and can deal with ambiguities, such as incomplete instructions. However, the instruction disambiguation focuses on finding the subtasks' sequence to perform when the main instruction does not explicitly describe all the required subtasks. In addition, this method does not consider disambiguation methods, such as spoken interaction, when the entities involved in performing the instruction are not precise.

In the present work, we aim to address some of the issues a service robot faces when it receives linguistic commands inside a home environment. We emphasize the importance of having common knowledge, dealing with missing information, and interacting by speech.

This work focuses on developing an effective system based on ontological knowledge that uses inference processes and spoken dialog to deal with missing information in commands. We intend to find a balance between the inferences and questions made in order to initiate natural interactions with humans. We perform experiments to compare human behavior and a robot that uses this system in a service robot domain.

3. System overview

The system is composed of four main modules with different functions. These modules manage the robot's knowledge and reasoning, command analysis, decision-making, and talking interaction. These modules are interconnected and share information between themselves in different steps of the processes. One of the modules has a link to the robot's perception and control modules, which enables the robot to perform the actions physically.

The system can perform some generic tasks, such as bringing an object, delivering an object to a place, finding an object, going to a place, taking an object, and placing an object. An overview of the system is shown in Fig. 1. General explanations of the modules and their functions are presented in the following subsections.

3.1. Knowledge management

The first part of this module is the main ontology, which contains knowledge about the objects in the environment and their relationships. For instance, we can instantiate an object named *mug* and create a statement indicating that it belongs to the class *Mug*. With this connection, the ontology can immediately relate the *mug* to similar superclasses, such as *Cup* and *DrinkingVessel*.

Using this ontology, we can assign properties to the instances of the objects it has, and define classes connected by property restrictions. Considering the object class *Mug*, we can assign the property *hasDefaultLocation*, which refers to a place where individuals of the class *Mug* can usually be found. The property *hasDefaultLocation* is linked to the class *Cabinet*, which is a *Furniture*. As a result, it is possible to deduce which furniture can be linked to a mug by the relation "*Mug hasDefaultLocation* some *Cabinet*".

This module contains the definition of the ontology and implements the formation of DL queries to access its knowledge. These DL queries are constructed as needed when a natural language command is received and during the information disambiguation process. Direct and indirect property values can be assigned to instances and classes using these queries. This module also makes complex inferences by querying links between the instances, properties, and classes, as explained in the subsequent sections.

3.2. Command interpretation

3.2.1. Command analysis

For a human, instructing a robot by speech is effortless and natural. One of the objectives of this module is to enable robots to receive commands through natural language and execute them.

This module processes a spoken command given by the human and creates goals based on the command. For instance, the human can tell the robot, "Go to the living room, take the soda, and place it on the coffee table". This module will generate the following set of goals for the command:

1. Go to the living room.
2. Find and grasp the soda.
3. Place the soda on the coffee table.

With this set of goals, the robot can begin the task execution process. However, the robot needs to verify whether some information is missing before accomplishing these goals. The missing information often depends on the actual environment setup, which could differ between houses. Some examples of possibly missing information from the previous set of goals are as follows:

- the location of the soda in the living room
- the type or name of soda, in case there is more than one
- the location of the coffee table
- which coffee table the human is referring to, in case there is more than one

The missing information differs according to the explicitness of the command. When the robot receives a command such as "Bring me the soda", it still needs to determine the location of the soda and the type of soda (if relevant). For this, the robot first creates DL queries based on the natural language command. The information obtained from that query is included in the generated goals. The clarification and disambiguation of this information are performed in the command refinement process that is part of the Task Planning and Execution module.

3.2.2. Talking interaction

This module is in charge of the interaction between the robot and the human. It generates questions according to the missing information to complete a task. It also validates whether the answer fulfills the robot's question. If necessary, the module restructures the question or asks a new one.

For the command "Bring me the soap", the robot can either think or interact to find the possible location of the soap. If the robot interacts, it may need to know the kind of soap using this module so that it can infer its location, or it can ask for the location directly. The robot might ask the following questions:

- Where is the soap?
- What kind of soap? Dishwashing? Laundry?

For instance, if the human response is "It is on the shelf", the robot might be confused as to which shelf in which room the human is referring to. By contrast, if the answer were "It is on the kitchen cabinet", the robot would directly go to the kitchen and approach the cabinet.

The next step for the robot is to determine the best question to ask by knowing its environment. The robot might encounter different situations. On the one hand, soaps could be kept inside a container in the laundry room. In that case, the robot needs to ask for the kind of soap the human wants, not its location.

On the other hand, dishwashing soap could be in the kitchen cabinet, and laundry soap next to the washing machine. In this case, both questions will be helpful. In case there is no usual place for the soap, the best option would be to ask where the soap is.

In summary, several situations arise depending on the environment setup. This module generates questions that will likely obtain the information the robot needs to fulfill a task.

3.3. Task planning and execution

This module generates a robot-specific subset of tasks based on the set of goals generated by the Command Analysis module. This subset describes each goal in a specific way such that the robot can execute them directly.

For example, for the first goal, "Go to the living room", the subtask generated would be sending the coordinates of the living room to the navigation function. Other goals may require more subtasks to be completed. For instance, when there is only one object called *soda*, the goal "Take the soda" would result in the following subtasks:

1. Receive the coordinates of the soda (from the previous goal, "Find the soda").
2. Check whether nothing is obstructing the path.
3. Grasp the *soda* located at the given coordinates.

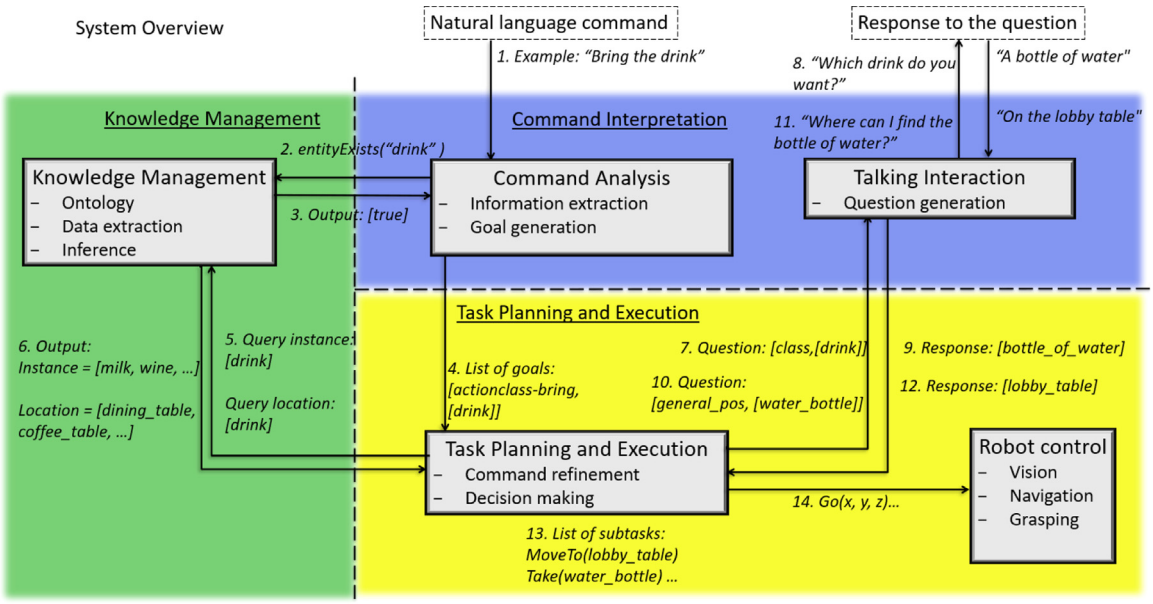

Fig. 1. System overview.

These are examples of the possible subtasks created. However, the number of generated subtasks depends on the information given from the subset of goals and thus might vary.

The information included in the set of goals received from the Command Analysis module might not be enough to execute the subtasks. This module queries the Knowledge Management module to acquire the missing information; if it stills need added information afterward, it interacts with the user. The information found during the inferring process is used to create questions for the user. The robot is not expected to face a new concept; instead, it will receive a request of a known object in an unclear manner.

4. Knowledge management

The organization and management of knowledge are an essential part of the proposed system. The robot needs to have a clear concept of the entities with which it interacts in its surrounding environment. We use an ontology to represent this knowledge. In this section, we explain the base ontology and then the inference processes along with their outputs, both corresponding to the Knowledge Management module.

4.1. Ontology

We use the standardized description language Web Ontology Language (OWL). OWL has a high degree of expression, thereby enabling the inference of complex implicit knowledge [43]. Different ontology models can be found for robots, depending on their application. We adopt an open-source ontology for robots included in the KnowRob framework that best match our purpose. The KnowRob framework is designed to provide knowledge to totally autonomous robots [21].

However, our system aims to use language-based human interaction to supplement the robot's knowledge and facilitate natural human–robot communication. KnowRob requires translation of predicates to query the ontology, which sometimes results in unnecessarily long and inefficient queries [44], and the interpretation does not consider human interaction to solve ambiguities. In our system, natural language commands are automatically translated into queries, and they are interpreted using ontological knowledge combined with spoken interaction with humans.

For these reasons, we utilize only the base ontology instead of the entire framework. The base ontology contains the conceptualization of household domains, which aims to provide vocabulary that describes events, objects, actions, states, and parameters. Although the base ontology contains well-described concepts, we add certain types of classes and object properties. We also instantiate the individuals representing the real objects in the environment and their respective associations.

We perform this adaptation to complete missing class names according to the description of our environment, and to facilitate querying classes that are frequently used. Due to these changes, we have to modify some object property names, change their data types, and adjust some of the relationships between them.

We use statements to describe the concepts (axioms) of classes and properties in the ontology. By using axioms, we can create and associate, for instance, classes, properties, restrictions, individuals, and intersections. An axiom can describe the connection between classes with a *SubClassOf* property by assigning a value to a data property, such as *integer*, *Boolean*, *float*, and other *data types*. It also describes the *domains* and *ranges* of properties and indicates whether an entity is equivalent to another entity, and other varieties of restrictions. The ontology management module reads this information and creates OWL axioms that are based on this information. These axioms are added to the ontology, whose consistency is then checked.

We supplement the ontology with instances of objects possibly found at home, as explained in the experimental scenario shown in Section 6. The Common sense regarding the objects' default location is established according to the house's layout in the experiments. This knowledge can be easily extended if necessary, as long as the concepts are appropriately described and associated. Moreover, it can be changed to describe new environments, such as hospitals and nursing homes.

4.2. Inference process

In OWL, inferences are used to access to concepts in the ontology. We rely on a so-called reasoner to access the information in the ontology using DL queries. A semantic reasoner is a piece of software that can infer logical consequences from a set of asserted facts or axioms. It helps reduce the redundancy of information and finds conflicts in knowledge content [45]. If

the ontology is not consistent, some conflicts or misconceptions might arise. Using the reasoner, we perform the query answering to find asserted statements, infer knowledge, check consistency, and identify hierarchical relationships.

We use the OWL API [46] for querying the information in the ontology. This library contains functions for working with OWL ontologies using the Java programming language and tools for DL parsing. We use this library to design methods of accessing the information stated in the ontology, perform query reasoning through DL queries, and extract the output information as needed.

Some of the reasoners' methods, the queries constructed during the command interpretation and disambiguation, are explained below.

Algorithm 1 getFirstTypeOfIndiv. It gets the name of the first type of class of an Individual.

Input: The name of the individual *indivName* and an indicator for considering just direct classes *isDirect*
Output: The name of the class *firstClassName*

```
1: indivInstance = getOneIndividual(indivName)
2: nodesetClasses = rsn.getTypes(indivInstance, isDirect)
3: entitiesList = nodesetClasses.entities()
4: firstType = entitiesList.findFirst()
5: firstClass = firstType.get()
6: firstClassName = getNameof(firstClass)
       return firstClassName
```

Algorithm 2 checkIndividualExists. It confirms if an Individual exists.

Input: The name of the individual *indivName*
Output: Whether the individual exist or not, *true* or *false*

```
1: listOfIndividuals = ontology.individualsInSignature()
2: entityExists = false
3: for each i ∈ listOfIndividuals do
4:     if getNameof(i) == indivName then
5:         entityExists = true
6:     end if
7: end for
8: return entityExists
```

Algorithm 3 getDataPropValuesOfIndiv. It gets the value from a Data property of an Individual.

Input: The name of the individual *indivName* and the name of the data property *dtPropName*
Output: Value of the property *value*

```
1: indInstance = getOneIndividual(indivName)
2: dtPropInstance = getOneDataProperty(dtPropName)
3: value = empty
4: aLiteral = rsn.dataPropValues(indInstance, dtPropInstance)
5: value = aLiteral.getValue()
       return value
```

Algorithm 1 receives the name of an individual and an indicator to return direct or potentially direct classes. It first brings the instance of the individual by using the input name on line 1. Then, it obtains the class types of the individual on line 2. Finally, it extracts the name of the first immediate class and return it (line 6).

Algorithm 2 confirms whether an *individual* exists inside the ontology by accessing the method included in the library to bring the list of individuals (line 1). It iterates (line 3), checks whether the given name matches the name of an *individual*, and returns whether it exists. This method is useful, for example, for checking whether the object that the user is asking for is instantiated in the robot's knowledge.

With Algorithm 3, we can access the value assigned to a *data property* of an *individual*. Since it receives the name of the individual and the data property to access, it first looks for the instance of those entities in the ontology (line 1). Then, it applies one of the reasoner methods to extract the values by sending the instances on line 4.

Not all the values of a property can be obtained immediately with the names and instances of the entities provided. Thus, we need to look through their connection with other entities as well.

We created additional general methods to access inferred knowledge through *ClassExpressions* with a DL query parser. These methods receive a DL query, which is parsed to an *OWLClassExpression*. Then, they extract the required entities using the reasoner. For instance, Algorithm 4 retrieves the list of *SuperClasses* that satisfies the given expression.

These algorithms are used at different steps of the command interpretation process to extract data and infer knowledge. The received linguistic command is preprocessed, as explained in Section 5, to extract the essential words from it and create query statements.

Ideally, the linguistic representation of the objects extracted from the command should be mapped to the individuals; this would require less inference process to know with certainty the objects needed from the real world. However, we consider cases where the user can ask for a specific target object (*individual*) or a general target object (*class*) in the command, e.g., "Bring the bottle of water" and "Bring a bottle".

In the first command, the target object can be mapped directly to an *individual bottle_of_water*. The second command refers to the class *Bottle*, in which case the system needs to find the specific individual to complete this action. For the second case, the linguistic representation can be mapped to a class first, which would require more inference process and a possible speech interaction with the commander.

Algorithm 4 getSuperClassesOfClassExpression. It gets the list of Super classes of a ClassExpression.

Input: The DL-query *classExpDL* and an indicator to clarify if they are directly stated classes or with potentially direct classes too *isDirect*
Output: A list of super classes *allSuperClassesList*

```
1: allSuperClassesList = empty
2: classExpParsed = parseDLExpression(classExpDL)
3: nodesetClasses =
   rsn.getSuperClasses(classExpParsed, isDirect)
4: entitiesList = nodesetClasses.entities()
5: for each cl ∈ entitiesList do
6:     tempClassName = getNameof(cl)
7:     allSuperClassesList.add(tempClassName)
8: end for
9: return allSuperClassesList
```

Some of the processes that utilize these algorithms are shown in Section 5. The entire list of methods of accessing the ontology can be found in Appendix B.

The following shows an input and output for the abovementioned algorithms. When the robot receives a command, such

as "Bring the milk", it sends the input = [milk] to Algorithm 2 to verify whether an object called *milk* exists inside the current environment, and it receives back *true* or *false*. Moreover, by sending the input = [milk, graspable] to Algorithm 3, it can verify whether the object *milk* can be grasped by the robot; for this property, it sends back *true* or *false*. For the other properties, it can return other kinds of values, such as *black* and *white* when color is queried.

5. Command interpretation and command refinement

The proposed system aims to interact with humans by receiving spoken commands and executing them. The robot must understand the meaning of the words and what these words refer to in each command. To accomplish this, the system transforms the command given in natural language into DL statements to query the ontological knowledge. The obtained information is used to generate a set of goals that are based on it. When uncertainties are found, the information is validated and refined first through alternative assertions in the ontological knowledge and inferring processes, and then through interaction with the human. In the following subsections, we explain how these goals are created.

5.1. Information extraction and grammar rules

In this subsection, the processes performed by the Command Analysis module are explained. The first step in analyzing the command is part-of-speech (POS) categorization, and dependency parses identification. With POS categorization or POS tagging, we can divide the words that share common grammatical properties and assign them to a class [47]. With the dependency parser, we obtain the type of relationship between the "head" words and their dependent words, which modify them. We implement Stanford Log-linear POS Tagger [48] and Stanford Parser [49], which are parts of the Stanford CoreNLP toolkit [50], for this purpose.

After the tagging and dependency parsing are completed, the system extracts relevant information to create a DL query which will help find the objects' instances satisfying the given concept description required for each task. The results of the query are passed as arguments to the next step.

Finally, the system identifies the *action classes* and their respective arguments, which are called *keywords*. We define an *action class* as a group of verbs sharing similar semantic properties.

Patterns of a particular group of verbs can be defined considering the lexico-syntactic structure of the verb phrase. For instance, the verb "go" is commonly followed by a noun that represents a place. Some verbs require more complex validation, such as double-object verbs [51]. Therefore, we can identify *keywords* for each *action class*. The *action classes* and their respective *keywords*, which represent objects or locations that can be identified, are defined in Listing 1. The inferred information is used to generate the corresponding *action class*.

Listing 1: Definition of Commands received by each Action Class.

⟨*actionclass-bring*⟩ ::= ⟨*verbs-bring*⟩ 'the' ⟨*object*⟩ ['from the' ⟨*location*⟩]

⟨*actionclass-deliver*⟩ ::= (⟨*verbs-deliver*⟩ | ⟨*verb-take*⟩) 'the' ⟨*object*⟩ 'to the' ⟨*furniture*⟩

⟨*actionclass-find*⟩ ::= ⟨*verbs-find*⟩ 'the' ⟨*object*⟩ ['in the' ⟨*location*⟩]

⟨*actionclass-go*⟩ ::= ⟨*verbs-go*⟩ 'to the' ⟨*location*⟩

⟨*actionclass-get*⟩ ::= ⟨*verbs-get*⟩ 'the' ⟨*object*⟩ ['from the' ⟨*location*⟩] | ⟨*verb-take*⟩ 'the' ⟨*object*⟩ ['from the' ⟨*location*⟩] ['to the' ⟨*furniture*⟩]

⟨*actionclass-place*⟩ ::= ⟨*verbs-place*⟩ 'the' ⟨*object*⟩ 'on the' ⟨*furniture*⟩

⟨*location*⟩ ::= ⟨*furniture*⟩ | ⟨*room*⟩

⟨*verbs-bring*⟩ ::= 'bring' | 'give me'

⟨*verbs-deliver*⟩ ::= 'deliver'

⟨*verbs-find*⟩ ::= 'find' | 'locate' | 'look for'

⟨*verbs-go*⟩ ::= 'go' | 'navigate' | 'enter'

⟨*verbs-get*⟩ ::= 'get' | 'grasp' | 'pick up'

⟨*verbs-place*⟩ ::= 'place' | 'put'

⟨*verb-take*⟩ ::= 'take'

⟨*furniture*⟩ ::= instance or class name of type furniture

⟨*room*⟩ ::= instance or class name of type room

⟨*object*⟩ ::= instance or class name of a graspable object

5.2. Goal and final subtasks generation

After the *action class* identification and *keyword* extraction, it is verified whether no more information is needed or whether the information is not precise, such as when the same concept description describes several objects. Then, with the information completely disambiguated, the set of goals can be generated. The disambiguation process is explained in Section 5.3.

After forming the set of goals and *keywords* in the Command Analysis module, the system needs to create a collection of final tasks that are based on each goal; the robot will subsequently execute these tasks. The generation of the final subtasks is performed by the Task Planning and Execution module.

As shown in Listing 1, an *action class* can receive different numbers of arguments, depending on how much detail the human provides in the command. Therefore, we establish an algorithm for each *action class* that receives different combinations of *keyword* patterns to specify the series of subtasks needed.

For instance, we create a method of selecting the valid entities required for the *action class Bring*; it uses the inferred keywords and disambiguates unclear data by interacting or querying related property statements, as shown in Algorithm 5, by which subtasks are generated (lines 25–28).

This algorithm describes the process of identifying the correct object and the specific source location to find it and bring it back to the commander. It starts receiving the *keywords* inferred from the natural language command, which contain the set of possible objects and source locations. The output of the given inference can be empty, which means the reasoning process did not find a suitable object and location according to the command description. This algorithm checks the cases where

- the object and the location are both found and unique (line 3),
- several objects correspond to the entire description (line 7),
- several objects and one source location correspond to the entire description (line 10),
- several objects and several source locations correspond to the entire description (line 13),
- no object corresponds to the entire description (line 16),
- there is an alternative location for the initial object when no other object matches the description (line 19), and
- there are several alternative source locations for the initial object when no other source location matches the description (line 22).

Once the concrete object and location are identified, then the robot can execute with certainty the rest of the commands included in this *action class* (lines 25–28), which are as follows:

1. Go to the place where the object is.
2. Take the object (including finding and grasping).
3. Go back to the place where the person is.
4. Hand over the object.

5.3. Command refinement

The Task Planning and Execution module performs the command refinement process explained in this subsection, which needs information from both the ontological knowledge and the user interaction to disambiguate the commands.

We explain some of the queries submitted to the Knowledge Management module (Section 5.3.1) and the generation of questions of the Talking Interaction module (Section 5.3.2), which help refine the user's command.

Communication in natural language can be complicated and ambiguous. When humans receive commands, they use different skills to understand them and be confident before doing them. Humans inadvertently access knowledge acquired from learned experiences and immediately know whether they have enough information.

In this part of the system, we attempt to enable the robot to mimic this human behavior by using ontological knowledge as a first attempt to understand the command and then interacting with humans to solve ambiguities in the command.

5.3.1. Ontology access

The robot can access the knowledge kept in the ontology at any time as it attempts to understand the command. However, there are cases where the inference process might not find an instance with the natural language command's initial description.

For instance, when the command requires a *book from the kitchen*, the knowledge will state that *books are usually in bookshelves*, and *bookshelves are not found in kitchens*. In this case, the initial query, which is based solely on *book from the kitchen*, will not retrieve any instance of a book.

Consequently, an alternative *furniture name* where to find the *book* is needed. The robot first queries the ontological knowledge to find an alternative location. Then, if no output is retrieved, the robot proceeds to interact with the user, as explained in the following subsection.

For the example of Algorithm 5, on line 18, a query for finding an alternative source location of the initial object is generated when no inferred possible objects are received.

The number of queries generated to find alternative instances of a concept description depends on the *action class* definition. For example, the *action class Deliver* needs an object and a destination, according to the definition shown in Section 5.1. This *action class* method will need to find an alternative source location and determine the correct destination if needed.

Table 1
Labels with semi-static questions and types of answers.

Label	Question	Answer type
aspect	What [property] is it?	Property description
class	Which [object_class] do you want?	Object name
general_pos	Where can I find the [object]?	Room or furniture
options_pos	Is it on the [furniture] in the [room_1] or in the [room_2]?	Room
specific_pos	In which piece of furniture can I find the [object]?	Furniture

5.3.2. User interaction

In regular conversations, humans omit details when they think they are unnecessary or to communicate faster. The same happens when humans give commands. However, the robot needs a way to know the omitted information.

The robot can find such information by using its ontological knowledge to infer those statements. Another way is to interact with the person who is giving the command. The robot interacts with the user for assistance when the general inference rules cannot find the appropriate answer.

In this part of the system, the robot can create questions based on the information it needs to complete its understanding of the tasks to be executed. In a question generation system, the use of domain knowledge helps create significant questions and vary specificity [52]. We apply the notion of generating questions from concepts where we can formulate flexible questions instead of having completely fixed ones.

We define the set of semi-static "Wh" questions shown in Table 1. The set includes labels identifying the types of questions and the strings describing them. Each question requires a word to be completed depending on the previously given task, and it expects specific information about the answers. The missing word and the expected answer can be the name of a graspable object, the object class, a property of the object, a room, or furniture.

Due to the broad information stored in the ontological knowledge base, we have the convenience of creating questions that add details, thus narrowing the answer expected from the user. The input for creating a question consists of two components: a label that describes the type of question to ask (e.g., position or object class) and a set of concepts that should be included in the question.

Consider the command "Bring me the salad bowl". Following the generation of the set of subtasks, the robot knows that it needs the furniture and the room. Then, the robot can use the input [general_pos, [salad bowl]], which will return the question "Where can I find the salad bowl?"

After receiving the answer from the human, the robot needs to confirm the location of the given position (e.g., the *kitchen cabinet* or the *high table* in the *kitchen*) or only the name of the room (e.g., the *kitchen*). Although the *kitchen* is received as the answer, the robot still needs to find the furniture inside the *kitchen* where the *salad bowl is*.

The robot can encounter more complex situations where the questions must include more concepts to narrow the answer. If the user asks, "Bring me the book", the robot can take advantage of its ontological knowledge to create a more precise question.

Let us consider a scenario with two shelves — one in the living room and one in the bedroom. By querying the ontology, the robot finds that books are usually on shelves. The robot can use this information as an input, [options_pos, shelf, [living room, bedroom]], and receive a question, such as "Is it on the shelf in the living room or in the bedroom?"

The response is narrowed to two options, unlike the possible responses to the question in the first example. Interacting with humans enables natural communication and allows the robot to

Algorithm 5 Break down the goal for the Action Class Bring, which takes the list of keywords inferred and the list of initial keywords.
Input: The list of inferred keywords *inferredKeywords* and initial keywords from the linguistic command *initialKeywords*

```
objsOptions = inferredKeywords.getObjects
sourceLocs = inferredKeywords.getSourceLocations
if objsOptions is 1 and sourceLocs is 1 then
    validObjN = objsOptions.getObject
    validSrcLocN = sourceLocs.getsourceLoc
end if
if objsOptions more than 1 then
    typeObj = objsOptions.getClass
    validObjN = callInteraction('class', typeObj)
    if sourceLocs is 1 then
        validSrcLocN = sourceLocs.getsourceLoc
    else
        validSrcLocN = callInteraction('general_pos', validObjN)
    end if
end if
if objsOptions not found then
    validObjN = initialKeywords.getObject
    alternativeLocs = makeDLquery(iObj, 'hasDefaultLocation')
    if alternativeLocs is 1 then
        validSrcLocN = alternativeLocs.getsourceLoc
    else
        validSrcLocN = callInteraction('specific_pos', iObj)
    end if
end if
moveToLocation(validSrcLocN)
takeObject(validObjN)
moveToLocation(initial_position)
handOverObject()
```

clarify the information before taking action. It also reduces the time needed by the robot to find the object.

By generating questions from concepts, we look for balance between the inference process and the questions made. The inferred knowledge gives us the advantage of selecting the pertinent question and including useful data about it. Moreover, we aim to avoid overusing the dialog to solve a task, which might lead to an unnecessarily large number of questions [38,41].

6. Experiments in simulated environment

We conducted two separate experiments inside a simulated environment to analyze human behavior and compare it with the behavior of a robot that used our system. Fig. 2 shows the outline of the experiments.

In the first experiment, the human subjects received commands that they had to fulfill. They could interact with the commander with no restrictions, as the communication must be as natural as possible.

In the second experiment, the same commands were given to the robot. It could use its knowledge, interact, or perform a combination of both to complete the tasks.

In these experiments, we can observe the different sequences of decisions followed by the subjects. The average solution for each command helped us discern what the subjects preferred to do to fulfill the same commands.

We investigated the combinations that led to possibly faster solutions. Furthermore, we used it as guidance to specify the range of an admissible number of decisions for each command. We analyzed and compared the results of both experiments, as shown in the following subsections.

We aim to show the feasibility of using inferred knowledge along with verbal interaction to solve tasks such as searching and finding objects, rather than looking for a system that will outperform humans.

6.1. Environment layout and contents

We designed a four-room house using the simulator Gazebo. The house is composed of a kitchen, bedroom, living room with dining room, and lobby (Fig. 3). Each room has static furniture, such as tables, cabinets, a sofa, and a bed. It also has graspable objects, such as food, dishes, drinks, toys, electronic devices, and clutter (Fig. 4). Most of the objects in the environment are organized intuitively, according to the house's layout. Food containers and kitchen utensils are in the kitchen, stationery on the office desk, and books on the bookshelf. However, there are some objects whose location might not always be the same, such as toys, tools, and personal items (e.g., wallet and cellphone). These objects are randomly located around the house and do not have a default location assigned.

We used a virtually modeled Toyota Human Support Robot (HSR) [53] as the service robot for these experiments. It has the same features and configurations as the real HSR.

We composed eight commands, as shown in Table 2. They included going and finding commands. Some did not contain explicit details about the place to go to, the place to start looking for the object, or the specific object to search. The purpose of having commands with missing information and objects with no default location was to motivate reasoning about the context and interaction with the instructor.

6.2. Experiment with humans in simulation

In this experimental scenario, the human subjects had to follow the commands shown in Table 2 inside the simulation. We

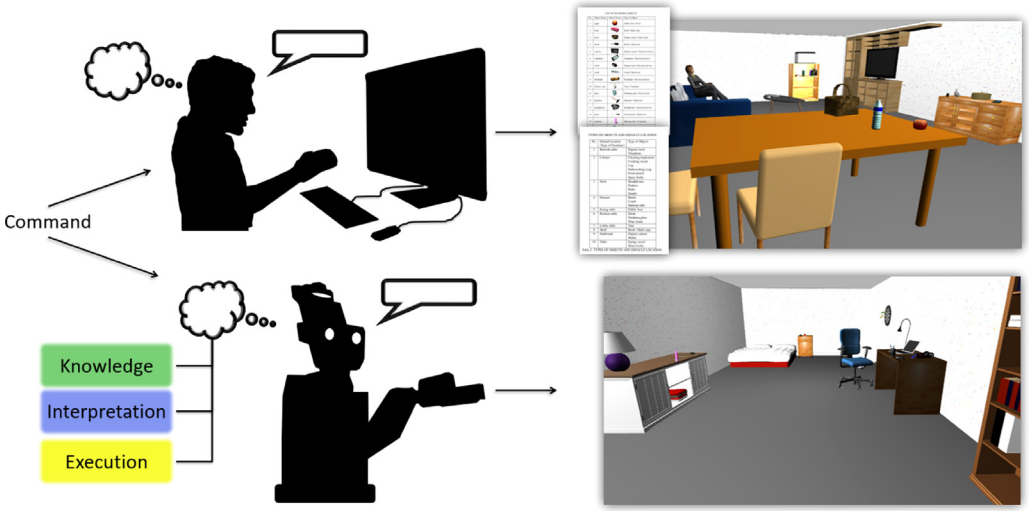

Fig. 2. Outline of the experiments. Top: Human subject receiving a command. The subject's view includes information about the objects in the simulated environment, and the camera view perspective. Bottom: Robot receiving a command. The objects' information is in the ontology, and the camera view has the same perspective as the human subject. The figure shows the camera view of the living room (top) and the bedroom (bottom).

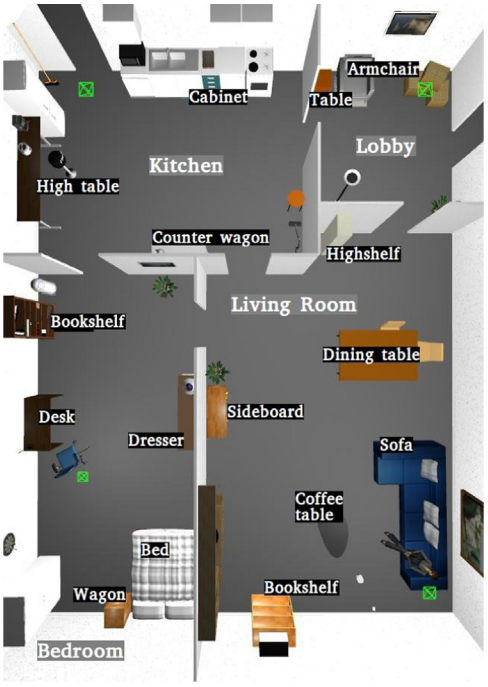

Fig. 3. Layout of the house in Gazebo simulator.

Fig. 4. Examples of graspable objects.

provided the subjects with a full list of the graspable objects, a list of the furniture, and a house map. The list of graspable objects included the name, picture, and type of the object. The list of furniture contained the type of furniture and the type of objects they usually store. The house map is shown in Fig. 3. During the experiment, the subjects were able to look through all three items at any time.

Table 2
Commands used in the experiments.

No.	Command
1	Go to the kitchen.
2	Find the bottle of wine in the kitchen.
3	Go to the living room and find a book.
4	Find a toy in the living room.
5	Find the makeup.
6	Find the Rubik's cube.
7	Find the hammer on the table.
8	Find a drink.

Each subject's camera view provided a first-person perspective; the controls for the subject's robot included moving forward, moving backward, and turning; they were allowed to go around freely through the house until they finished the task.

We selected eight people who could speak English (basic to fluent) to fulfill each of the eight commands, resulting in a total of 64 natural language commands completed. Some of the subjects had previous experience with service robots and simulations. This diversity of the subjects provided contrasting conditions; the robot-experienced subjects were accustomed to similar simulations, but their English-speaking skills might lead to varied responses.

Measuring the performance of a service robot in a real environment is difficult. Some robotics competitions use independent test sets to benchmark robots' performance and abilities [54]. Such benchmarking is divided into system benchmarking and component benchmarking, where the system is evaluated as a whole and by a single functionality, respectively. The design of these benchmarks considers functional abilities, such as navigation, person recognition, mapping, speech recognition, and object manipulation. It also takes into account system properties, such as ease of use, calibration speed, ergonomics, and adaptivity [54,55]. As service robots evolve and the environment changes, benchmarks need to be continuously improved, and new ones might appear [56].

However, these benchmarks mostly evaluate an entire system's or task's success or failure rather than assessing the way the

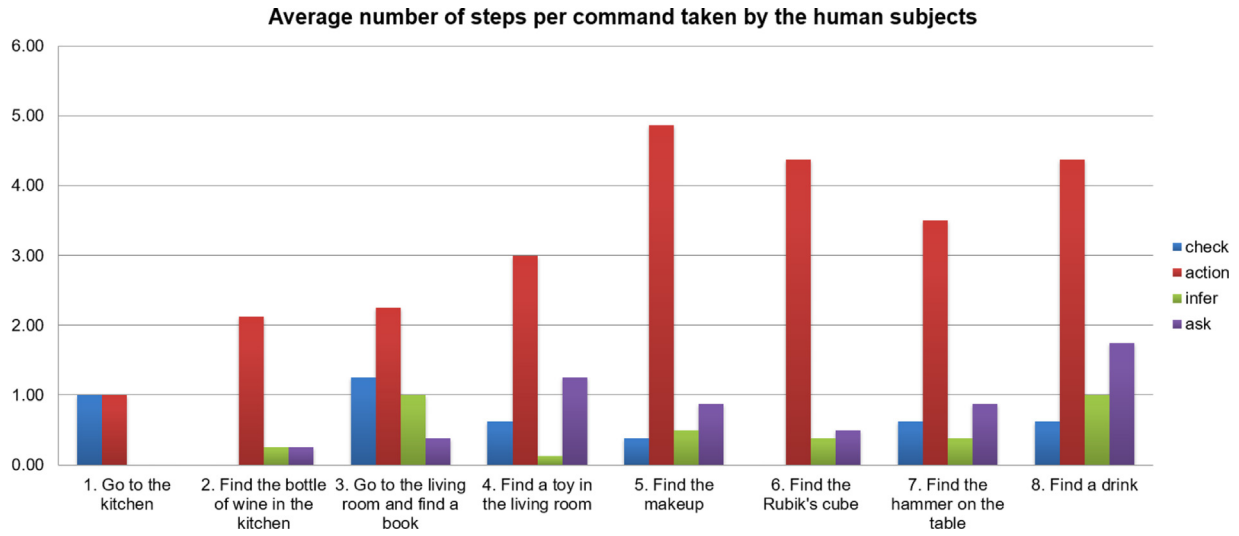

Fig. 5. Average number of steps per command taken by the human subjects.

task is completed. Our target is to observe the different variations of decisions made to complete the commands in the experiments. For these reasons, we decided to estimate each subject's performance by the number of times they interacted with the instructor by asking questions, checked information about the environment, and thought and acted inside the simulation. To measure these data, we asked the subjects to verbally describe what they were doing or thinking during the experiment.

We counted every action (step) performed by the subjects that is, every time

- (ask) the subject interacted with the instructor, e.g., asked or confirmed information;
- (check) the subject checked information about the environment, e.g., location of furniture, class of an object;
- (infer) the subject made some inference, e.g., the food must be in the kitchen; and
- (act) the subject performed an action inside the simulation, e.g., moving from one room to another.

Examples of the steps taken by a human subject for the command "Find a drink" are listed in Table B.4.

Fig. 5 shows the average number of steps taken for each command. The graph shows that the predominant step is acting, whereas the others differ depending on the command. For instance, for the fifth command ("Find the makeup"), most of the subjects preferred to take action and attempted to find the *makeup* instead of accelerating the process by asking. However, most of the subjects were unsure about the appearance or location of the *makeup*, resulting in a large number of actions. The total number of steps could have been reduced by combining asking and checking information.

For the third command ("Go to the living room and find a book"), the combination of information checking, inferring, and asking resulted in fewer steps. Moreover, a *book*'s location is more intuitive to assume compared with other objects, such as a *Rubik's cube* or a *hammer*.

Regarding the seventh and eighth commands ("Find the hammer on the table" and "Find a drink"), the subjects needed all their skills (e.g., checking information, making inferences) to find the *hammer* and the specific *drink* correctly. In this environment, we did not assign a specific location for the *hammer*; thus, it could be found anywhere. Moreover, this scenario contained several drinks, such as *milk*, *wine*, and *water*. Hence, the last command required skills other than acting to complete it.

An interesting observation arose in these experiments. We expected the fluent English speakers to interact more with the instructor or create better questions, resulting in better performance. However, we noticed that the subject's English-speaking skills did not interfere significantly. Some subjects who were not highly fluent performed better in certain commands. Their decisions may have been affected by their personalities rather than language skills.

6.3. Experiment with robot in simulation

To test our system, we performed the experiment using a virtual HSR with our proposed system as the robot's mind. The robot received the same set of commands as the humans did. It had the option of thinking (accessing the ontological knowledge), interacting, or performing a combination of them before executing the tasks.

We determined its performance with the same level of definition as that in the experiments with the humans. A step was recorded every time the robot interacted, checked information, made an inference, or executed an action inside the simulation.

Fig. 6 shows the number of steps that the robot made per command. The different combinations of steps and the total combinations per command are detailed. For most of the selected cases, accessing the information in the ontological knowledge base was essential. However, for the second command ("Find the bottle of wine in the kitchen"), the combination of acting, inferring, and asking was optimal for understanding and execution.

Some commands could be completed in a few steps, while others required a larger number of steps due to different reasons. For instance, the second command already included certain information; thus, the robot needed only a few steps to complete the missing data.

The third and fourth commands ("Go to the living room and find a book" and "Find a toy in the living room") had a similar structure; both of them included the object class and location. The main difference was in the actual scenario; by making an inference, the robot could find the furniture where the *books* were. By contrast, to know the name and location of the *toy*, the robot needed to ask the instructor more questions.

The eighth command ("Find a drink") needed a larger number of steps. The robot needed access to ontological knowledge to check for objects that matched *drink*. It needed inference processes to know the possible location for the *drink* according to this

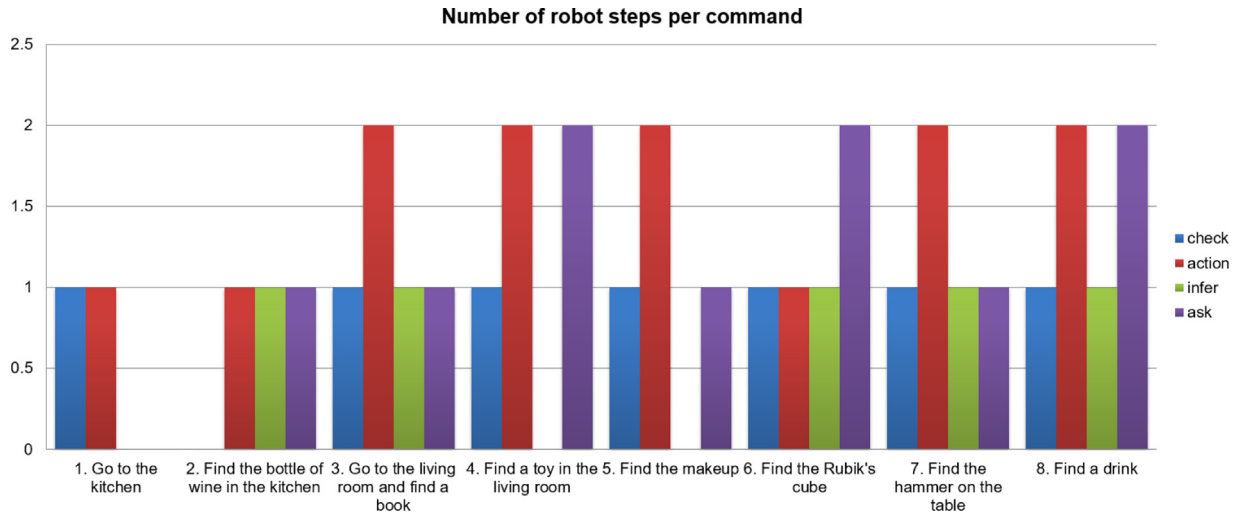

Fig. 6. Number of steps per command of the robot.

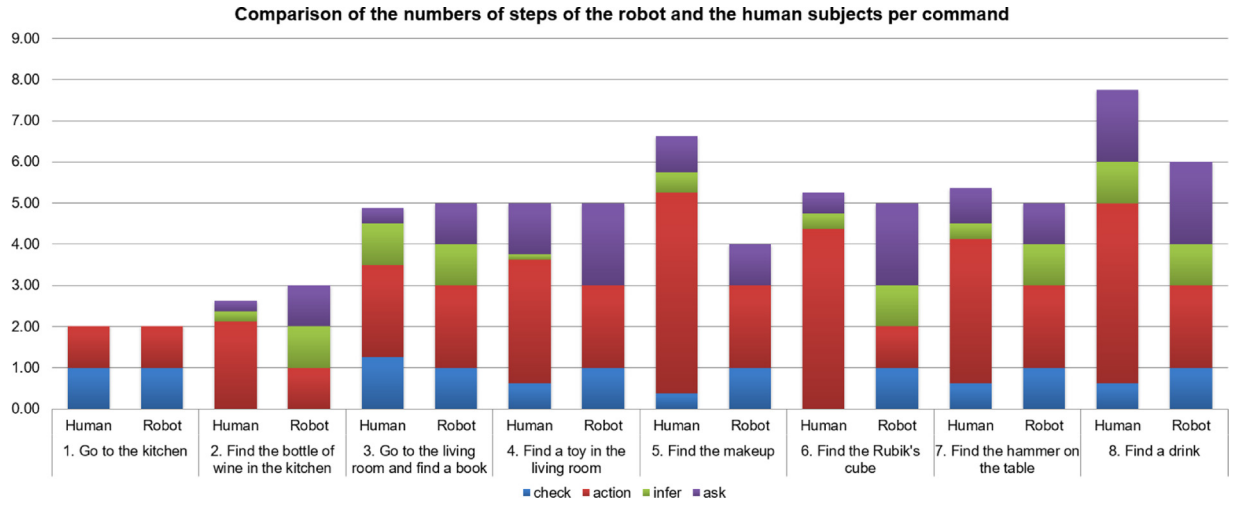

Fig. 7. Comparison of the numbers of steps of the robot and the humans subjects per command.

environment. The robot also had to interact with the commander to know which *drink* they needed to find.

The robot attempted to balance its skills to acquire all the information needed to accomplish the tasks. Similarly, in the fourth and eighth commands ("Find a toy in the living room" and "Find a drink"), the robot benefited from its skill of interacting when its knowledge was insufficient.

6.4. Comparison and analysis of experiments

To determine whether our proposed system is adequate for such service robot tasks, we compared the results obtained in the abovementioned experiments. Fig. 7 compares the average numbers of human and robot steps per command. Although the numbers of steps for some commands are similar, their combinations are different for almost all cases.

For the second command ("Find the bottle of wine in the kitchen"), the humans made fewer inferences but more actions than the robot, thus completing the task faster. However, for the fifth command ("Find the makeup"), the robot made fewer steps by checking for information in the ontological knowledge for this scenario.

Regarding the sixth command ("Find the Rubik's cube"); the total numbers of steps did not considerably differ. The humans preferred to do more actions and less reasoning, whereas the robot combined all of its skills. Measuring the results for this command through the number of steps might not yield a significant difference. However, in a real situation, acting requires more time than reasoning; thus, the humans took longer than the robot to complete this task.

Fig. 8 compares the maximum and minimum numbers of steps taken by the humans and the robot for each command. The minimum number of steps may be deemed the best combination, since the more steps the humans took, the longer the time they needed to complete a task. The numbers of steps by the robot in all cases were within the minimum and maximum. For the fifth and eighth tasks, the robot's scores were close to the minimum, which meant the robot's performance was excellent.

The system showed competitive performance compared to the human subjects in the experimental setup. However, this might change if the experiments are performed in a real environment. The human subject's behavior could differ if they interact in an environment that is natural for them; hence, their performance could improve.

Knowledge management is an essential component of the proposed system; this was evaluated through the times the robot accessed stated and inferred knowledge. The human subjects' knowledge was similarly assessed; we observed during the experiments every time they used the environment's information

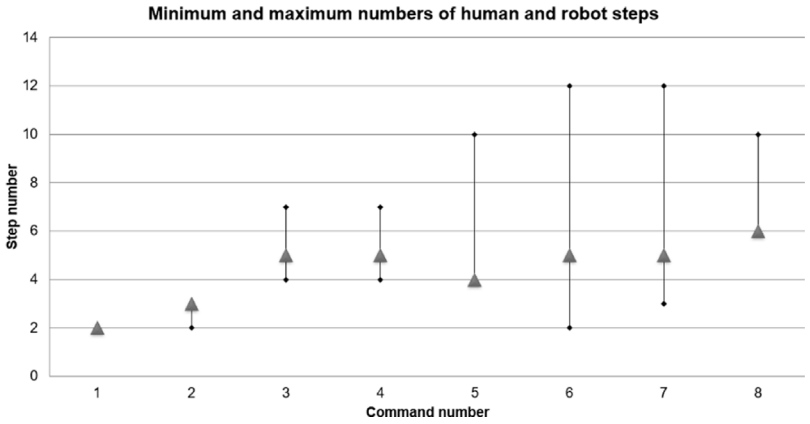

Fig. 8. Minimum and maximum numbers of human and robot steps.

Table A.3
List of domain-general methods.

Type	Name	Description
Data reasoning	getAllClassesofIndividual	Brings a list of the classes that an individual belongs to and their superclasses
	getDataPropValuesOfIndiv	Brings the value from a data property of an individual
	getEquivalentClassesOfClassExpression	Brings a list of the equivalent classes that describe a given class expression; it can be specified to return directly stated or potentially direct
	getFirstTypeOfIndividual	Brings the name of the first type of class of the given individual
	getIndividualsOfClass	Brings a list of individuals that belong to the specified class
	getInstancesOfClassExpression	Brings a list of individuals that satisfy the class expression
	getObjectPropertyofIndividual	Brings the value from an object property of an individual
	getPropValuesOfIndiv	Brings the value from a property of an individual; it does not need to specify the type of property
	getSuperClassesOfClassExpression	Brings a list of the classes that describe a given class expression; it can be specified to return directly stated or potentially direct
	individualBelongsToClass	Verifies if the given individual belongs to the specified class
Data access	checkExistsClass	Verifies if the given class exists
	checkIndividualExists	Verifies if the given individual exists
	containsADataProperty	Verifies if the ontology has the specified data property
	containsADataType	Verifies if the ontology has the specified data type
	containsAnIndividual	Verifies if the ontology has the specified individual
	containsAnObjectProperty	Verifies if the ontology has the specified object property
	entityExists	Verifies if the given entity exists
	getClassesName	Brings a list of the names of all the classes
	getFirstInstanceOfClass	Brings the first instance found of the given class
	getImmediateSuperClassOf	Brings the first superclass that the given class belongs
	getIndividualsName	Brings a list of the names of all the individuals
	getNameofClassasOntology	Brings the name of the given class, as defined in the ontology
	getOneClass	Brings the requested class
	getOneDataProperty	Brings the requested data property
	getOneDataType	Brings the requested data type
	getOneIndividual	Brings the instance of the requested individual
	getOneObjectProperty	Brings the requested object property
	getOwlDatatype	Brings the requested instance of OWLDatatype
	getOwlLiteralofDatatype	Brings an OWLLiteral with the specified value and data type
	getRangeOfDataProperty	Brings the range of the given data property
	getSuperClassesOfClass	Brings the list of superclasses that the given class belongs

given beforehand to check the data and infer facts. However, the subjects' knowledge would be more challenging to evaluate in a real-world experiment when the environment is utterly familiar to them; it is not known with certainty what knowledge they already have or what kind of inferences they would make.

7. Conclusions and discussion

This paper describes an ontology-based knowledge management system with verbal interaction for command interpretation and execution by home service robots. We propose the combination of ontological knowledge reasoning and human–robot interaction to interpret natural language commands. The system is composed of four main modules that (a) manage the robot's knowledge and reasoning, (b) analyze the command to generate goals, (c) refine the information and execute tasks, and (d) interact with humans by speech to disambiguate information. The system relies on inference methods and verbal interaction to understand commands and clarify uncertain information.

We tested the proposed system in a scenario where the robot received commands, such as going to rooms and finding objects, with missing or unclear information. It had to understand them by using reasoning and interaction to be able to execute them. We performed another experiment where human subjects solved the same set of commands. We then compared the performance of the system and the human subjects.

Some improvements could be achieved in future work, such as the interpretation of natural language statements with entirely

Table B.4
Sample list of steps taken by a human subject for the command "Find a drink." The numbers, types, and descriptions of the steps are shown.

Step No.	Step type	Step description
1	Check	The subject accesses the environment information and checks which objects are drinks. Result: The subject finds milk, water, and wine.
2	Ask	The subject asks, "Any drink?" The commander replies: "A bottle of water please."
3	Infer	The subject thinks and decides that the first place to look at is the table in the lobby. Note: The subject was near the lobby when the command was received.
4	Act	The subject approaches the lobby table. Result: There is no water bottle on the table.
5	Ask	The subject asks: "Do you know where I can find it?" The commander replies, "in the living room."
6	Check	The subject checks the map information to find a table. Result: The subject finds the dining table.
7	Act	The subject goes to the living room.
8	Act	The subject approaches the dining table. Result: The subject finds the bottle of water.

new entities. To accomplish the mentioned skill, we need to implement a new sequence of actions to acquire a new concept and new dialog-based clarification methods to handle inconsistencies in ontological knowledge.

The concepts represented in the ontology can be easily extended with more home-related concepts or a new environment definition. The linguistic representation of verbs associated with the current action class description can also be extended. However, increasing the keywords involves modifying the goal generation's disambiguation process; adding new action classes requires the definition of new methods describing their main sequence. These bring an opportunity to adapt the system to create the subtasks sequence dynamically.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the Japan Society for the Promotion of Sciences (JSPS) KAKENHI under Grant 17H01799.

Appendix A. List of methods of accessing the ontology.

This appendix contains the list of methods developed to access the ontology at different levels. The methods are classified as domain-general and task-specific methods.

The domain-general methods are then divided into data reasoning and data access (Table A.3). The data reasoning methods help obtain information that is not necessarily stated in the ontology and thus needs to be deduced from the assertions. The data access methods retrieve information that is directly stated in the ontology and can access any property of it.

Appendix B. Sample list of steps taken by a subject.

This appendix contains a sample list of the steps taken by a human subject during the experiments. The subject received the command "Find a drink". The list shows the step number, type, and description, along with the result of the execution.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2021.103763. The video presents three examples of commands performed by the robot and a human subject during the experiments. The presented cases include their behavior when the robot had a (1) bad, (2) good, and (3) equal performance compared to the human subject.

References

[1] S. Nielsen, E. Bonnerup, A. Hansen, J. Nilsson, L. Nellemann, K. Hansen, D. Hammershøi, Subjective experience of interacting with a social robot at a danish airport, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), in: IEEE RO-MAN proceedings, IEEE, United States, 2018, pp. 1163–1170, http://dx.doi.org/10.1109/ROMAN.2018.8525643.

[2] B. Bruno, C.T. Recchiuto, I. Papadopoulos, A. Saffiotti, C. Koulouglioti, R. Menicatti, F. Mastrogiovanni, R. Zaccaria, A. Sgorbissa, Knowledge representation for culturally competent personal robots: Requirements, design principles, implementation, and assessment, Int. J. Soc. Robot. 11 (3) (2019) 515–538, http://dx.doi.org/10.1007/s12369-019-00519-w.

[3] H.M.R.T. Bandara, M.A.V.J. Muthugala, A.G.B.P. Jayasekara, D.P. Chandima, Grounding object attributes through interactive discussion for building cognitive maps in service robots, in: IEEE International Conference on Systems, Man, and Cybernetics, SMC 2018, Miyazaki, Japan, October 7-10, 2018, 2018, pp. 3775–3780, http://dx.doi.org/10.1109/SMC.2018.00639.

[4] C. Li, G. Tian, H. Chen, The introduction of ontology model based on SSO design pattern to the intelligent space for home service robots, in: 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2016, pp. 1673–1678, http://dx.doi.org/10.1109/ROBIO.2016.7866568.

[5] H. Jeon, K. Yang, S. Park, J. Choi, Y. Lim, An ontology-based home care service robot for persons with dementia, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 540–545, http://dx.doi.org/10.1109/ROMAN.2018.8525668.

[6] S. Chandra, R. Paradeda, H. Yin, P. Dillenbourg, R. Prada, A. Paiva, Do children perceive whether a robotic peer is learning or not?, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, in: HRI '18, ACM, New York, NY, USA, 2018, pp. 41–49, http://dx.doi.org/10.1145/3171221.3171274.

[7] A. Bajcsy, D.P. Losey, M.K. O'Malley, A.D. Dragan, Learning from physical human corrections, one feature at a time, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, in: HRI '18, ACM, New York, NY, USA, 2018, pp. 141–149, http://dx.doi.org/10.1145/3171221.3171267.

[8] A. Angleraud, Q. Houbre, V. Kyrki, R. Pieters, Human-robot interactive learning architecture using ontologies and symbol manipulation, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 384–389, http://dx.doi.org/10.1109/ROMAN.2018.8525580.

[9] D.P. Losey, M.K. O'Malley, Enabling robots to infer how end-users teach and learn through human-robot interaction, IEEE Robot. Autom. Lett. 4 (2019) 1956–1963.
[10] T. Horie, K. Takashio, Handling conversation interruption in many-to-many HR interaction considering emotional behaviors and human relationships, in: RO-MAN 2018 - 27th IEEE International Symposium on Robot and Human Interactive Communication, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 528–533, http://dx.doi.org/10.1109/ROMAN.2018.8525745.
[11] S. Patki, A.F. Daniele, M.R. Walter, T.M. Howard, Inferring compact representations for efficient natural language understanding of robot instructions, 2019, arXiv:1903.09243.
[12] R. Liu, X. Zhang, A review of methodologies for natural-language-facilitated human–robot cooperation, Int. J. Adv. Robot. Syst. 16 (3) (2019) 1729881419851402, http://dx.doi.org/10.1177/1729881419851402, arXiv:https://doi.org/10.1177/1729881419851402.
[13] M. Serna, A. Serna, Ontology for knowledge management in software maintenance, Int. J. Inf. Manage. 34 (2014) 704–710.
[14] A. Eckhard, I. Navas Delgado, J. Aldana Montes, Towards an ontology for enterprise knowledge management, in: SEMAPRO International Conference on Advances in Semantic Processing, 2011, pp. 75–80.
[15] F. Ali, D. Kwak, P. Khan, S.H.A. Ei-Sappagh, S.M.R. Islam, D. Park, K. Kwak, Merged ontology and SVM-based information extraction and recommendation system for social robots, IEEE Access 5 (2017) 12364–12379, http://dx.doi.org/10.1109/ACCESS.2017.2718038.
[16] J. Zhang, W. Zhao, G. Xie, H. Chen, Ontology- based knowledge management system and application, Procedia Eng. 15 (2011) 1021–1029, http://dx.doi.org/10.1016/j.proeng.2011.08.189, CEIS 2011.
[17] J.I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, M. Habib, A. Khamis, A. Olivares, E.P. de Freitas, E. Prestes, S.V. Ragavan, S. Redfield, R. Sanz, B. Spencer, H. Li, Ontology for autonomous robotics, in: 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2017, pp. 189–194, http://dx.doi.org/10.1109/ROMAN.2017.8172300.
[18] D. Beßler, M. Pomarlan, M. Beetz, OWL-Enabled assembly planning for robotic agents, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, in: AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2018, pp. 1684–1692.
[19] G.A.G. Ricardez, Y. Osaki, M. Ding, J. Takamatsu, T. Ogasawara, Estimating the operation of unknown appliances for service robots using CNN and ontology, in: 2018 Second IEEE International Conference on Robotic Computing (IRC), 2018, pp. 181–182.
[20] B. Fan, J. Ma, N. Jiang, H. Dogan, R. Ali, A rule based reasoning system for initiating passive ADAS warnings without driving distraction through an ontological approach, in: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2018, pp. 3511–3517.
[21] M. Tenorth, M. Beetz, Knowrob: A knowledge processing infrastructure for cognition-enabled robots, Int. J. Robot. Res. 32 (5) (2013) 566–590, http://dx.doi.org/10.1177/0278364913481635, arXiv:https://doi.org/10.1177/0278364913481635.
[22] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mösenlechner, D. Pangercic, T. Rühr, M. Tenorth, Robotic roommates making pancakes, in: 2011 11th IEEE-RAS International Conference on Humanoid Robots, 2011, pp. 529–536, http://dx.doi.org/10.1109/Humanoids.2011.6100855.
[23] M. Beetz, M. Tenorth, J. Winkler, Open-EASE, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1983–1990, http://dx.doi.org/10.1109/ICRA.2015.7139458.
[24] M. Tenorth, D. Nyga, M. Beetz, Understanding and executing instructions for everyday manipulation tasks from the world wide web, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 1486–1491, http://dx.doi.org/10.1109/ROBOT.2010.5509955.
[25] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. Bozcuoglu, G. Bartels, Know rob 2.0 — A 2nd generation knowledge processing framework for cognition-enabled robotic agents, 2018, pp. 512–519, http://dx.doi.org/10.1109/ICRA.2018.8460964.
[26] A. Ramachandran, C.-M. Huang, E. Gartland, B. Scassellati, Thinking aloud with a tutoring robot to enhance learning, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, in: HRI '18, ACM, New York, NY, USA, 2018, pp. 59–68, http://dx.doi.org/10.1145/3171221.3171250.
[27] J. de Wit, T. Schodde, B. Willemsen, K. Bergmann, M. de Haas, S. Kopp, E. Krahmer, P. Vogt, The effect of a robot's gestures and adaptive tutoring on children's acquisition of second language vocabularies, in: Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, in: HRI '18, ACM, New York, NY, USA, 2018, pp. 50–58, http://dx.doi.org/10.1145/3171221.3171277.
[28] Z. Materna, M. Kapinus, V. Beran, P. Smrž, P. Zemčík, Interactive spatial augmented reality in collaborative robot programming: User experience evaluation, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 80–87, http://dx.doi.org/10.1109/ROMAN.2018.8525662.
[29] Y. Cheng, Y. Shi, Z. Sun, D. Feng, L. Dong, An interactive scene generation using natural language, in: 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 6957–6963, http://dx.doi.org/10.1109/ICRA.2019.8794327.
[30] B. Galitsky, D. Ilvovsky, E. Goncharova, On a chatbot conducting dialogue-in-dialogue, in: Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue, Association for Computational Linguistics, Stockholm, Sweden, 2019, pp. 118–121.
[31] M. Abrams, L. Gessler, M. Marge, B. Rex: a dialogue agent for book recommendations, in: Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue, Association for Computational Linguistics, Stockholm, Sweden, 2019, pp. 418–421.
[32] M.J. Chung, M. Cakmak, "How was your stay?": Exploring the use of robots for gathering customer feedback in the hospitality industry, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 947–954, http://dx.doi.org/10.1109/ROMAN.2018.8525604.
[33] L. Shu, P. Molino, M. Namazifar, H. Xu, B. Liu, H. Zheng, G. Tür, Flexibly-structured model for task-oriented dialogues, 2019, ArXiv abs/1908.02402.
[34] H. Sugiyama, T. Meguro, Y. Yoshikawa, J. Yamato, Improving dialogue continuity using inter-robot interaction, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 105–112, http://dx.doi.org/10.1109/ROMAN.2018.8525542.
[35] N. Axelsson, G. Skantze, Modelling adaptive presentations in human-robot interaction using behaviour trees, in: Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue, Association for Computational Linguistics, Stockholm, Sweden, 2019, pp. 345–352.
[36] E. Sibirtseva, D. Kontogiorgos, O. Nykvist, H. Karaoguz, I. Leite, J. Gustafson, D. Kragic, A comparison of visualisation methods for disambiguating verbal requests in human-robot interaction, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), IEEE, 2018, http://dx.doi.org/10.1109/roman.2018.8525554.
[37] K. Wolfel, D. Henrich, Grounding verbs for tool-dependent, sensor-based robot tasks, in: 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2018, pp. 378–383.
[38] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, J. Hart, P. Stone, R.J. Mooney, Improving grounded natural language understanding through human-robot dialog, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, http://dx.doi.org/10.1109/icra.2019.8794287.
[39] E. Bastianelli, G. Castellucci, D. Croce, R. Basili, Textual inference and meaning representation in human robot interaction, in: Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora, Trento, Italy, 2013, pp. 65–69, URL https://www.aclweb.org/anthology/W13-3820.
[40] E. Bastianelli, D. Croce, A. Vanzo, R. Basili, D. Nardi, A discriminative approach to grounded spoken language understanding in interactive robotics, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, in: IJCAI'16, AAAI Press, 2016, pp. 2747–2753.
[41] J. Bos, T. Oka, A spoken language interface with a mobile robot, Artif. Life Robot. 11 (2007) 42–47, http://dx.doi.org/10.1007/s10015-006-0397-5.
[42] D.K. Misra, J. Sung, K. Lee, A. Saxena, Tell me dave: Context-sensitive grounding of natural language to manipulation instructions, Int. J. Robot. Res. 35 (1–3) (2016) 281–300, http://dx.doi.org/10.1177/0278364915602060, arXiv:https://doi.org/10.1177/0278364915602060.
[43] Knowledge representation in description logic, in: Semantic Web: Concepts, Technologies and Applications, Springer London, London, 2007, pp. 35–55, http://dx.doi.org/10.1007/978-1-84628-710-7_3.
[44] S. Lee, I. Kim, A robotic context query-processing framework based on spatio-temporal context ontology, Sensors 18 (10) (2018) http://dx.doi.org/10.3390/s18103336, URL https://www.mdpi.com/1424-8220/18/10/3336.
[45] S. Abburu, Article: A survey on ontology reasoners and comparison, Int. J. Comput. Appl. 57 (17) (2012) 33–39, Full text available.
[46] The OWL api, 2019, http://owlcs.github.io/owlapi/ (Accessed: 2019-12-16).
[47] Words, parts of speech, and morphology, in: An Introduction To Language Processing with Perl and Prolog: An Outline of Theories, Implementation, and Application with Special Consideration of English, French, and German, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 113–145, http://dx.doi.org/10.1007/3-540-34336-9_5.

[48] K. Toutanova, D. Klein, C.D. Manning, Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, in: NAACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, 2003, pp. 173–180, http://dx.doi.org/10.3115/1073445.1073478.

[49] D. Chen, C.D. Manning, A fast and accurate dependency parser using neural networks, in: EMNLP, 2014.

[50] C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard, D. McClosky, The stanford corenlp natural language processing toolkit, in: Association for Computational Linguistics (ACL) System Demonstrations, 2014, pp. 55–60, URL http://www.aclweb.org/anthology/P/P14/P14-5010.

[51] R.K. Larson, On the double object construction, Linguistic Inquiry 19 (3) (1988) 335–391, URL http://www.jstor.org/stable/25164901.

[52] M.M. Berg, A. Isard, J.D. Moore, An openccg-based approach to question generation from concepts, in: E. Métais, F. Meziane, M. Saraee, V. Sugumaran, S. Vadera (Eds.), Natural Language Processing and Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 38–52.

[53] Y. Ota, Partner robots – from development to business implementation, in: Z.S. Hippe, J.L. Kulikowski, T. Mroczek (Eds.), Human – Computer Systems Interaction: Backgrounds and Applications 2: Part 2, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 31–39, http://dx.doi.org/10.1007/978-3-642-23172-8_3.

[54] T. Wisspeintner, T. Zant, L. Iocchi, S. Schiffer, Robocup@home: Scientific competition and benchmarking for domestic service robots, Interact. Stud. 10 (2009) 392–426, http://dx.doi.org/10.1075/is.10.3.06wis.

[55] D. Holz, L. Iocchi, T. Zant, Benchmarking Intelligent Service Robots through Scientific Competitions: the RoboCup@Home approach., AAAI Spring Symposium - Technical Report, 2013.

[56] M. Basiri, E. Piazza, M. Matteucci, P. Lima, Benchmarking functionalities of domestic service robots through scientific competitions, KI - Künstl. Intell. 33 (2019) http://dx.doi.org/10.1007/s13218-019-00619-9.

Liliana Villamar Gómez received the B. Eng. Degree in Computer Systems Engineering from Universidad del Valle de México, México, and M. Eng. Degree in Information Engineering from Toyohashi University of Technology, Japan, in 2014 and 2018, respectively. She is currently a Ph.D. candidate in the Department of Computer Science and Engineering, Graduate School of Engineering, Toyohashi University of Technology, Japan. Her research interests include service robots, human–robot interaction, artificial intelligence, and language processing.

Jun Miura received the B.Eng. Degree in Mechanical Engineering in 1984, the M. Eng. and the Dr. Eng. Degrees in Information Engineering in 1986 and 1989, respectively, all from the University of Tokyo, Tokyo, Japan. In 1989, he joined the Department of Computer-Controlled Mechanical Systems, Osaka University, Suita, Japan. Since April 2007, he has been a Professor at the Department of Computer science and Engineering, Toyohashi University of Technology, Toyohashi, Japan. From March 1994 to February 1995, he was a Visiting Scientist at the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. He received several awards, including Best Paper Award from the Robotics Society of Japan in 1997, Best Paper Award Finalist at ICRA-1995, and Best Service Robotics Paper Award Finalist at ICRA-2013. Prof. Miura published over 200 papers in international journals and conferences in the areas of intelligent robotics, mobile service robots, robot vision, and artificial intelligence.