Applying Semantic Web Technologies to Multi-Robot Coordination

Zhengjie Fan[1], Elisa Tosello[2], Michele Palmia[3], and Enrico Pagello[2]

[1] INRIA & LIG
655, avenue de l'Europe, Montbonnot Saint Martin
38334 Saint-Ismier, France
zjfanster@gmail.com

[2] Intelligent Autonomous Systems Lab (IAS-Lab)
Department of Information Engineering (DEI)
University of Padova
{toselloe,epv}@dei.unipd.it
http://robotics.dei.unipd.it

[3] micpalmia@gmail.com

Abstract. In this paper we introduce a framework that uses Semantic Web Technologies to facilitate multi-robots coordination. These Technologies let the creation of a Knowledge Base supporting the assignment of tasks to robots according to their capabilities. We address two problems: data sets interlinking and robots coordination. We interlink data sets to avoid repeated queries and we improve the coordination as follows. While usually a fixed coordination algorithm is used; in our approach, our Knowledge Base contains a set of algorithms and the tools able to select the most appropriate one according to the characteristics of the task.

Keywords: Semantic Web, Robotics, Coordination

1 Introduction

For many years, robots were designed to work in isolation performing complex tasks in high precision applications, such as industrial manufacturing and service robotics. Nowadays purpose is to make robots user-friendly: they are required to interact with humans and other robots. An automata has, for example, to assist people in daily life. And a multi-robots system is expected, for example, to search victims under disasters' ruins. For this reason, making robots autonomous and intelligent is a challenge. Giving robots autonomy guarantees their independence from humans' guidelines; giving them intelligence ensures their adaptation to changing environments. Nowadays, Semantic Web Technologies (SWT) are entering in the robotics scenario and they appear to be a powerful innovative instrument to achieve robots autonomy and intelligence.

According to the World Wide Web Consortium (W3C)[4], "the Semantic Web provides a common framework that allows data to be shared and reused across

[4] http://www.w3c.com/

application, enterprize, and community boundaries". It builds models for knowledge from different domains, in such a way that the concepts, relations and instances of these domains can be understood and processed automatically by software agents, without people's mediation [11]. Semantic Web Services are services described through the Semantic Web technologies [15]. Implementing Semantic Web Services upon a group of cooperating robots creates a network of distributed services able to semantically communicate. Such a network shares virtual data between robots and towards their coordination centers employing Semantic Web technologies to interpret sent and received messages. The vision is to achieve a time synchronization and a consequent execution coordination. In this way new tasks can be distributed using algorithms for the real-time reconfiguration, failure compensation and execution optimization. And there are several resulting advantages, among others the following ones. Humans are less involved, and robots' interaction is simpler: they can share actions' experiences in an easier way with a lower computational burden.

There are several research initiatives trying to make robots autonomous and intelligent by applying Semantic Web Technologies. For example, Ha et al. [8] developed a software package that uses Semantic Web technologies and AI-based planning techniques to provide automatic interoperation between networked robots and other computing devices: the Service-oriented Ubiquitous Robotic Framework (SURF). Other applications exist. Among them, KnowRob [14] and RoboEarth [16] are spreading in the robotics community. KnowRob is a knowledge processing framework developed to store Semantic data and to do reasoning for robots. It allows robots to execute high level tasks with very little a priori knowkedge, enabling easy sharing of action and object specifications. Robots equipped with such a system are referred to as knowledge-enabled. Moreover, robots that are obtaining knowledge and sharing knowledge with their teammates could use it to execute action specifications in a collaborative and coordinated manner. KnowRob is build over the Robot Operating System (ROS)[5]. It provides tools for knowledge acquisition, representation and reasoning. It allows to combine multiple knowledge sources. It supports heavy customization on what kind of knowledge is represented and on how the inference process is carried on. RoboEarth is a platform that "allows any robot with a network connection to generate, share and reuse data" on the Web. The RoboEarth project addresses the reuse and sharing of knowledge among autonomous robots, with the goal of fostering code and data reuse among different machines. But it is not clear how to express robots' capacities and how actions are allocated according to robots' compliances. To remedy such a lack, Juarez A. developed RoboDB [9, 10], a database that is able to represent robots' characteristics and abilities in a structured and sustainable way. Such a database allows the creation of complex and dynamic plans according to robots' capacities. Tasks can be assigned to robots capable to cooperate one with each other to fulfill the tasks.

This paper describes our goal to use KnowRob and RoboEarth to develop a framework that is able to coordinate a network of robots. Usual systems assign

[5] http://www.ros.org

tasks to robots according to a fixed coordination algorithm. The purpose of the present work is to create a system able to exploit the semantic Knowledge Base to select the most appropriate algorithm for each specific task. Moreover, we discuss the problems related to data sets' interlinking and forecast some solutions to avoid repeated queries.

The paper is organized as follows. In Section 2, an overview of our proposed framework is illustrated, together with the evaluation used to choose the implementation language. In Section 3, each module of the framework is described, especially the ones that contain the interlinking and the coordination problems. In Section 4 we expose some challenges emerged during the work and Section 5 contains the conclusions.

2 System Overview

Our framework is shown in Fig. 1. In the figure, Step 1 consists in building a local database that describes the information of the robots' working environment and capabilities. The database is built with Semantic Web languages according to ontologies from KnowRob, RoboDB or the ontologies we build on our own. Afterwards in Step 2, we should build links across the data sets in the local database, so as to facilitate data extraction. For example, robots' observations should be interlinked into a map of the working environment. In Step 3, each robot's capability is matched with the precondition and postcondition of a task, or the precondition and postcondition of each step of a task. A coordination algorithm is selected according to the matching result. In Step 4, an action plan can be made according to the matching result of Step 3 and the chosen coordination algorithm. Step 5 takes each action of the plan to be executed by robots. If there are other actions that have not been executed, the robot should observe the working environment again (Step 6), update the map and plan again (Step 7). If there is no action left, the task finishes successfully.

In order to build our framework, we first have to choose the appropriate Semantic Web languages to express knowledge in the local database. Three languages are compared: Extensible Markup Language (XML), Resource Description Format (RDF), and Web Ontology Language (OWL).

XML [4] is a language whose scope is creating a generic format to be served, received, and processed on the web. One of its pros is flexibility: the vocabulary of tags and their allowed combinations are not fixed and can be defined ad hoc for a certain application [5]. A con, other than its verbosity, is that data exchange requires a set of basic rules to allow different systems to communicate and understand each others. Encoding the information using XML requires then the use of an XML Schema [13]: a document containing the set of rules describing the structure of a XML document. RDF [1] is a data model for making statements on resources like triples of the form <subject-attribute-object>. RDF is a model, it does not need to be redefined when new knowledge should be stated: its schema stays the same. In fact, the RDF Schema (RDFS) only describes how to use RDF to describe RDF vocabularies. OWL [3] is a language

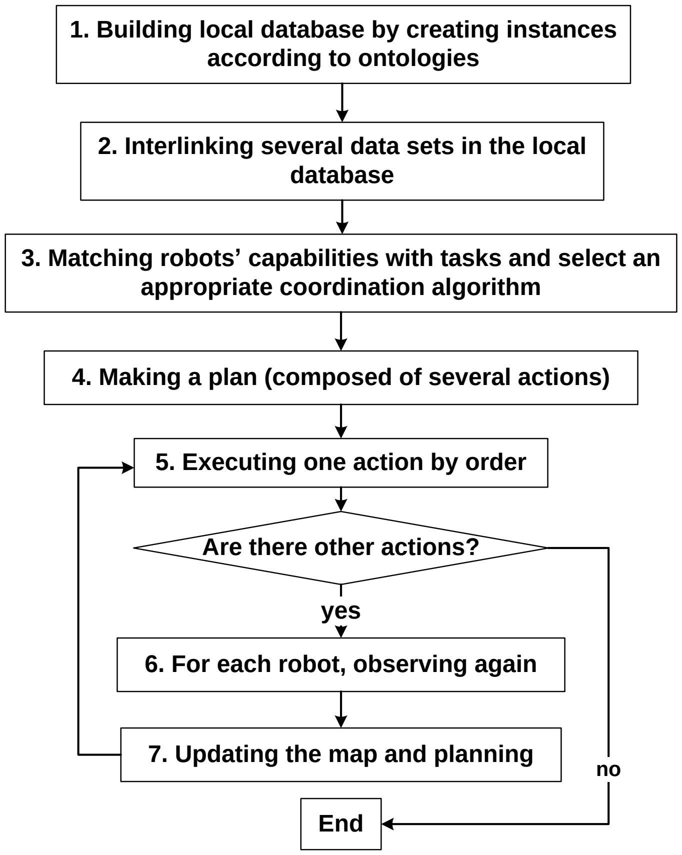
1. Building local database by creating instances according to ontologies
2. Interlinking several data sets in the local database
3. Matching robots' capabilities with tasks and select an appropriate coordination algorithm
4. Making a plan (composed of several actions)
5. Executing one action by order
Are there other actions?
yes
6. For each robot, observing again
7. Updating the map and planning
no
End

Fig. 1. Work Flow of the Framework

whose scope is that of facilitating machine interpretability of web content and effectively extends RDFS by providing additional vocabulary along with formal semantics. The disadvantage of encoding knowledge using RDF and OWL is the loss of expression flexibility and the inefficient reasoning comparing to XML: information has to be appropriately modeled with predefined vocabularies, and reasoning will be more difficult to be executed if more expressive vocabularies are applied.

After having compared the pros and cons of every language format, both RDF and OWL have been adopted. In such a way the resulting system lets the user to perform reasoning over the encoded data other than being the format used to encode the majority of the software packages and tools available.

3 Proposed Solutions of the Framework

Before talking about how to fulfill a task according to the framework of this paper, we first present a scenario as an example to clarify the following sections.

Assume two robots are asked to fulfill a moving task: move a table from one room to another room in a house. The work flow is: Search for the table → Hold the table → Move the table from one room to the destination room → Put the table down to the floor.

3.1 Step 1: Creating Instances According to Ontologies

When a group of robots is commanded to fulfilled a specific task in an environment, they need a database to store all data that they acquire in the working environment. Based on such a database, robots can communicate with each other through updated information. The schemas of the database are structured as Ontologies of Semantic Web. RoboDB and KnowRob both introduce ontologies for various kinds of tasks, environments and robots. With regard to these ontologies, we can build a local database for a network of robots.

For the scenario, the local database should store the map of the house, positions of the robots and positions of the table. These information are represented as instances of ontologies that are predefined. Furthermore, some instances should be updated during the moving task. Positions of the robots and of the table are both changed. Consequently, the instances of the corresponding positions should be updated.

Ideally, the following ontologies could be used:

1 An ontology that defines the concepts and relations describing the objects and their physical locations in the house. Such an ontology should apply a coordinate system that helps describing the geographical locations of each object in the house.
2 An ontology that defines the concepts and relations describing the relative distance between objects in the house. The concepts could be “near”, “far”, “beside”, “above” and etc. These concepts are defined with respect to the physical distances between objects. For example, if two objects’ distance is smaller than 1 meter, their relation can be defined as “near”.

For the ontology No.1 and No.2, an algorithm should be defined in order to infer the relative relation between two objects. An algorithm that computes distances of geographical locations between objects should also be designed. By referring to the relative relations of other objects, we can infer the relative relations between two objects that have changed their positions or whose relative relation has not been defined. For example, we do not know the relative distance between a robot and the table. If their distance is 10 meters and there is no “near” relation whose distance is equal to or larger than 10 meters, we can assume that the robot is “far” from the table.

Ontologies of KnowRob and RoboDB Fig. 2 shows the work flow of KnowRob. In the center of the figure, there is a knowledge database related to robot's tasks. The knowledge is expressed with Semantic Web languages. The knowledge comes from the observations of robots, interactions with human beings and online acquisition. The robot can do reasoning on the knowledge in order to extract useful information during its actions. KnowRob provides a series of predefined ontologies for creating a database. There are ontologies that describe actions, maps of the working environment and objects. We can make use of them to build the database for our framework.

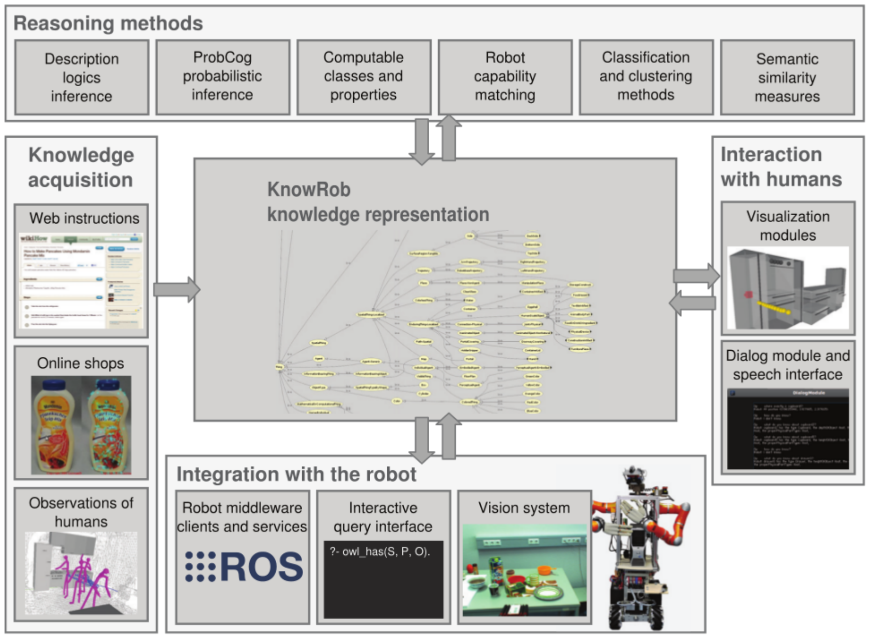
Reasoning methods
Description logics inference
ProbCog probabilistic inference
Computable classes and properties
Robot capability matching
Classification and clustering methods
Semantic similarity measures
Knowledge acquisition
Web instructions
Online shops
Observations of humans
KnowRob knowledge representation
Interaction with humans
Visualization modules
Dialog module and speech interface
Integration with the robot
Robot middleware clients and services
ROS
Interactive query interface
?- owl_has(S, P, O).
Vision system

Fig. 2. The KnowRob Work Flow

RoboDB also provides a series of predefined ontologies for us to build the database. These ontologies describe the body structure and features of robots.

3.2 Step 2: Interlinking Data Sets

Once instances of each ontology are created, we can maintain a local database that stores the useful information of fulfilling the moving task. However, it is better to interlink instances of different data sets, so that we do not need to query information from each data set. For example, one robot in the scenario observes one room of the house, where the table stays. The other robot observes the destination room, where the table should be moved to. The data set of instances that describes the objects in the first room should be interlinked with the one that describes the objects in the destination room. Among the two sets, there are instances that describe the same resource in the world, which is the door that connects the two rooms. By interlinking the two instances that describe the door, the two data sets can be treated as one set when extracting information with query languages of Semantic Web, such as SPARQL [2].

Interlinking can be done manually, if there are not many instances in both data sets. Otherwise, an algorithm should be applied to automate the interlinking process. In [6, 7], an interlink pattern of two data sets is built to compare instances and generate links across two data sets. Although the interlinking method also requires interactions with users for the sake of the interlinking precision, the computations of comparing instances are largely reduced than the ones of manually interlinking.

Algorithm 1 Interlinking Instances across Data Sets

```
Input: Two Data Sets
Output: Links across Data Sets
 1: The data set $D$, $D'$; /*two data sets to be interlinked*/
 2: Similarity threshold $T$;
 3: for (Each property/relation in the data set $D$) do
 4:   for (Each property/relation in the data set $D'$) do
 5:     Match properties/relations that are corresponding to each other and store as
        the alignment $A$;
 6:   end for
 7: end for
 8: for (Each instance in the data set $D$) do
 9:   for (Each instance in the data set $D'$) do
10:     Compare instances' property values according to the correspondences of the
        alignment $A$;
11:     Aggregate all similarities between property values as a similarity value $v$;
12:     if ($v >= T$) then
13:       The two compared instances are interlinked with owl:sameAs.
14:     end if
15:   end for
16: end for
```

Algorithm 1 aims at interlinking instances across two data sets D and D'. The algorithm first computes property/relation correspondences across two data sets (line 5). Then, instances' property values are compared by referring to the correspondences (line 10). A similarity value v is generated upon all similarities of property values (line 11). If such a similarity is equal to or larger than a predefined threshold T, the two compared instances can be used to build a link with the relation *owl*:*sameAs* (line 12-14).

An interlinking example is given in Fig. 3. In the figure, *http://www.task.it/ROOM1*, *http://www.task.it/ROOM2*, *http://www.task.it/DOOR1*, *http://www.task.it/DOOR2*, *http://www.task.it/DOOR_no1*, *http://www.task.it/TABLE* and *http://www.task.it/GARDEN* are URIs of seven instances *ROOM1*, *ROOM2*, *DOOR1*, *DOOR2*, *DOOR_no1*, *TABLE* and *GARDEN*. *connectObject*, *containObject* and *owl:sameAs* are relations that connect instances. Specifically, *owl:sameAs* is a relation that is defined to connect instances that refer to the same resource in the world. *color* and *size* are two properties of instances. Hence, *white*, *brown*, *15* and *20* are property values of instances.

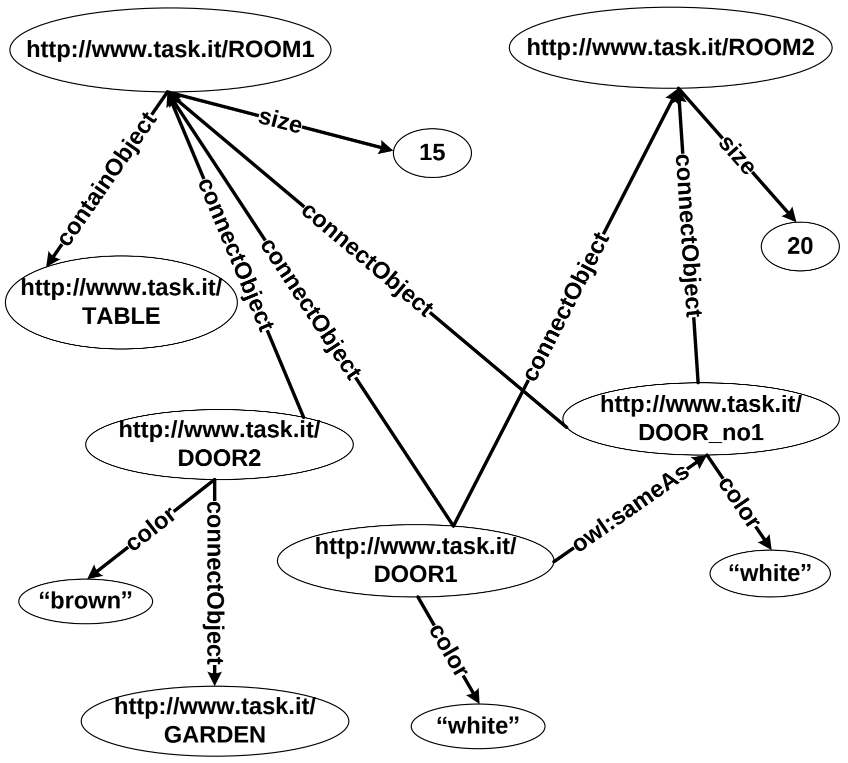

Fig. 3. The RDF Graph of All Objects Related to the Moving Task

Assume we are going to interlink several instances that are created according to each robot's observation towards one room. For the room *ROOM1*, there are two doors. One door *DOOR2* connects *ROOM1* with the garden *GARDEN*. The other door *DOOR1* connects *ROOM1* with *ROOM2*. For the room *ROOM2*, there is only one door *DOOR_no1*. It connects *ROOM1* and *ROOM2*. Obviously, *DOOR_no1* and *DOOR1* are referring to the same door. One reason is that they have the same color. The other reason is that they connect the same two rooms. Therefore, they can be interlinked with the relation *owl:sameAs*.

3.3 Step 3: Selecting Coordination Algorithm by Matching Robots' Capabilities with Tasks

Nowadays, the most common approach is allowing the robots team having only one coordinate algorithm that helps it to fulfill a fixed task. Such a working configuration is not flexible enough to utilize robots in different working environments. It is also a burden for the robot designers to write countless coordinate algorithms, in order to let the robot work in each specific task. If we assign several coordinate algorithms for a robots team, robots can be used in many working environments. For example, a robots team may have one coordinate algorithm for moving objects and another coordinate algorithm for detecting the environment by walking around. Such a flexibility should be supported by a selection algorithm, otherwise robots do not know which algorithm to execute when being asked to do a task.

Semantic Web Services can help us to design the selection algorithm. Different from the Semantic Web Service-based system depicted by Mokarizadeh et al. [12], we can treat each coordination algorithm as a Semantic Web Service. Each coordination algorithm has two kinds of information: input and output. They can be matched against tasks' features and the maps of the working environment, so that we are able to evaluate whether such an algorithm is able to help completing the task.

According to the scenario in this section, both robots match their capabilities against the task when they receive the command to move the table. Their capabilities can be defined by creating an instance of the concept "capability" with some property values. For example, the property "HoldingMaxWeight" can be set by 10kg. The matching process can be executed by comparing the property values of such an instance with the ones of the task's instance, which is also described by several property values. For example, "HoldingAnObject" can be set by the instance of the table. By checking the weight of the table, each robot can evaluate whether they are able to hold the table independently or with assistance.

Assume there are two coordination algorithms. The first one is executed when the tasks can be divided into several sub-tasks, each of which is fulfilled by one robot independently. The second one is executed when the task cannot be divided, so that the robots should work together. If the robots both can bear the weight of the table, we can ask only one of them to move the table. An algorithm of computing the execution cost is called. The execution cost is a function of several factors, such as the execution time. Accordingly, the robot which stays in the same room (the room *ROOM1*) with the table will take less time to move the table than the one stays in the other room (the room *ROOM2*). At this moment, no coordination algorithm is called, in that one robot is enough to fulfill the task. A sequence of actions is planned, which is "Hold the table"$\rightarrow$"Walk to the other room"$\rightarrow$"Put down the table". The chosen robot then walks to the table and holds the table. Nevertheless, after holding the table, it discovers that the table is not away from the floor. So it asks for help. At this point, a coordination algorithm should be called. Since the moving task cannot be divided into several sub-tasks, the second coordination algorithm is called. Then, the second robot is demanded to walk to the room *ROOM1*. Two robots hold the table and move it to *ROOM2* together. Furthermore, if there are also other coordination algorithms, it would be efficient to select an algorithm devoted to the coordination of robots during the holding and another devoted to the coordination of robots during the walking. The former will calculate the center of gravity of the object to be transported, the consequent total force needed to raise it, and the resulting force that each robot must apply to keep the object in balance during the transport. The latter will calculate the path that each robot has to follow in order to reach the goal position without colliding with the nearby objects. To conclude, selecting a coordination algorithm for each task encourages code reuse and avoids the creation of ad hoc algorithms.

Querying Sharing Knowledge on RoboEarth RoboEarth is a platform for robots to share data, action experiences and codes. We can query on the platform of RoboEarth in order to extract useful actions of similar tasks to fulfil robots' tasks. There are two proposed perspectives to search for similar tasks. The former aims to match similar actions of tasks on RoboEarth and the ones of robots' task; e.g., we can associate the tasks "Moving a table" and "Moving a chair", because they both have the action "move". The latter aims to match similar maps of tasks on RoboEarth and the ones of robots' task; e.g., we can associate the tasks "Moving a chair onto a table" and "Putting a ball on top of a chair", because they both change the map of the working environment to a map in which one object is on the top of another.

3.4 Step 4: Making a Plan

The plan of a task has to be formulated according to different selected robots' states and the selected coordination algorithm. A database like RoboDB is useful to collect the robots according to their characteristics. Then we should assign the actions that compose the plan to robot according to the matching result of Step 3 and the coordination algorithm. To improve the efficiency of the assignment, it is reasonable to have an estimation of the costs of each robot action (such as time and workload) and to choose an optimal action distribution strategy.

3.5 Step 5: Executing the Actions

No actual code nor any package is provided directly with the KnowRob system in order to implement action specifications' execution. The robot first has to check whether its current software and hardware assets correspond to what is needed to complete the task (via SRDL). Then, every action can be recursively decomposed into several atomic actions. Every atomic action has to be transformed into actual robot movements. This is done by using ROS and its *actionlib* library.

3.6 Step 6 and 7: Observing the Working Environment Again, Updating the Map and Planning

The solutions of these two steps are the same with the creation of instances according to the observations of robots in Step 1, interlinking instances in Step 2 and planning the actions in Step 4.

4 Challenges

There are still some challenges when we realize such a framework.

The first challenge is providing an expert (human or virtual) able to supervise robots trying to fulfill the assigned task. A decision system of this type can guide the selection and revision of the ontologies to be used, or it can enable robots to autonomously agree on the coordination or collaboration algorithm to be used.

The second challenge is choosing the appropriate ontologies for the demanding task. It is not an easy thing to choose the appropriate ontologies that can be used to construct the database of the working environment. The reason is that designers have different understandings on the concepts and relations. For example, young Chinese probably understands the concept “family” to be composed of the concepts “mother”, “father” and “one child”. Obviously, it is not the only family structure throughout the world. Hence, it is not easy to find out ontologies that fit in a specific task.

It is normal that there is no ontology that fits exactly in the requested task. Most of the time, we should do some revisions on the ontologies that are available on the web, so as to let them cover all needed concepts and relations of the robots’ working environment. If there is no appropriate ontology available on the web, we should design on our own.

5 Conclusions

In this paper we presented a framework that is able to coordinate a multi-robots system using SWT. Most of the existing semantic frameworks suffer from interlinking problems. Data sets are distributed on the robots’ network, but they are not connected with each other. Consequently, repeated queries are performed. We propose a method that properly interlinks instances that refer to the same resource in the world, so that all robots in the network share data with each other. Moreover, most of the existing semantic frameworks assign tasks to robots according to a fixed coordination algorithm. In our approach, a set of algorithms has been designed to select the most appropriate coordination algorithm according to the characteristics of the task. Based on this, the robots can be used to fulfill different tasks with the same set of coordination algorithms.

In our future work, we will realize the framework as the illustration of this paper. We will also make use of a specific task, such as “moving the table” to evaluate our framework. Since there are challenges presented in Section 4, an expert (human or virtual) that is able to supervise the system will be deployed. Its presence will not alter the autonomy of the robots network. It will only suggest behavioral improvements when necessary. As consequence, we will design the ontology that better fits the requirements of the working environment. The ontology will combine the requirements of the robotics behavior, together with that of the expert of the system.

References

1. Resource description format (rdf), `http://www.w3.org/RDF`
2. Simple protocol and rdf query language (sparql), `http://www.w3.org/TR/sparql11-query/`
3. Web ontology language (owl), `http://www.w3.org/TR/owl-features/`
4. Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F.: Extensible markup language (xml) 1.0 (fifth edition). Tech. rep., W3C Recommendation (2008)
5. Decker, S., Van Harmelen, F., Broekstra, J., Erdmann, M., Fensel, D., I., H., Klein, M., Melnik, S.: The semantic web: the roles of xml and rdf. IEEE Internet Computing 4(5), 63–74 (2000)
6. Fan, Z.: Concise Pattern Learning for RDF Data Sets Interlinking. Ph.D. thesis, University of Grenoble (2014)
7. Fan, Z., Euzenat, J., Scharffe, F.: Learning concise pattern for interlinking with extended version space. In: Accepted by The 2014 IEEE/WIC/ACM International Conference on Web Intelligence (2014)
8. Ha, Y., Sohn, J., Cho, Y.: Service oriented integration of networked robots with ubiquitous sensors and devices using the semantic web services technologies. In: Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on,. pp. 3947–3952. IEEE (2005)
9. Juarez, A., Bartneck, C., Feijs, L.: Using semantic technologies to describe robotic embodiments. In: Proceedings of the 6th International Conference on Human-robot Interaction. pp. 425–432. HRI '11, ACM, New York, NY, USA (2011), `http://doi.acm.org/10.1145/1957656.1957812`
10. Juarez Cordova, A.: Semantic web for robots: applying semantic web technologies for interoperability between virtual worlds and real robots. Ph.D. thesis, Eindhoven University of Technology (2012)
11. Matthews, B.: Semantic web technologies. Tech. Rep. 8, CCLRC Rutherford Appleton Laboratory, JISC (2005)
12. Mokarizadeh, S., Grosso, A., Matskin, M., Kungas, P., Haseeb, A.: Applying semantic web service composition for action planning in multi-robot systems. In: 2009 Fourth International Conference on Internet and Web Applications and Services. pp. 370–376 (2009)
13. Sperberg, McQueen, C., Thompson, H.: Xml schema, `http://www.w3.org/XML/Schema`
14. Tenorth, M., Beetz, M.: Tknowrob – a knowledge processing infrastructure for cognition-enabled robots. International Journal of Robotics Research (IJRR) 32 (2013)
15. Törmä, S., Villstedt, J., Lehtinen, V., Oliver, I., Luukkala, V.: Semantic web services - a survey. Tech. rep., Helsinki University of Technology, Laboratory of Software Technology (2008)
16. Waibel, M., Beetz, M., D'Andrea, R., Janssen, R., Tenorth, M., Civera, J., Elfring, J., Glvez-Lpez, D., Hussermann, K., Montiel, J., Perzylo, A., Schiele, B., Zweigle, O., van de Molengraft, R.: Roboearth - a world wide web for robots. Robotics & Automation Magazine 18 (2011)