A Conceptual Approach for Knowledge Structure Update and Learning in Multi-Agent Systems

EGONS LAVENDELIS
Department of Artificial Intelligence and Systems Engineering
Riga Technical University
1 Kalku Street, Riga, LV-1658
LATVIA
egons.lavendelis@rtu.lv

Abstract: - The paper proposes a concept for a knowledge representation approach that consists of the knowledge structure in the form of ontology, knowledge base in the form of rules and environment model in the form of objects in the environment. The approach is developed for domains where the main problem is to choose the most appropriate capability for a particular action. Both the agents' knowledge and also knowledge structures can be edited by the user after launching the system. An Ontology Learning Tool is implemented for this purpose. Additionally the knowledge base can be complemented by the agents themselves as a result of the learning process that is based on the user's feedback. The proposed approach is explained in this paper on an example of vacuum cleaning multi-robot system. The implementation is done in the form that is suitable for development of JADE based multi-agent systems. The first validation of the concept is intended in a virtual multi-agent environment where JADE agents simulate cleaning robots.

Key-Words: - Knowledge Structure, Multi-Agent System, Ontology Learning, Machine Learning

1 Introduction

Nowadays one of the widely used forms to structure knowledge and to define a common semantics in the communications among agents is the use of ontologies as a common framework to define the meanings of the concepts used. Agent development environments offer ontology support to enable knowledge sharing among agents by enabling to encode class instances into messages, for example JADE Ontology Support [2] allows encoding classes defined in the ontology into message content field of the FIPA ACL [6] messages. This approach fulfils the needs for semantics based communications. At the same time, it does not allow to change the ontology itself to obtain new concepts during the lifetime of the system.

Ontology learning is a new discipline that tries to solve the problem that the designer of the system has to create the knowledge structure in the form of ontology before the agents can use it. Various methods for automatically building ontologies and as a result saving human efforts exist. Authors of [5], [8] and [10] give good overviews of the existing methods. As concluded in [12], there are still many unsolved things in the area of ontology learning. Methods for learning ontologies from text and big data based methods dominate the state of the art [10]. Despite saving human efforts these methods do not change the situation that ontology must be ready before starting to use the system. Semi-supervised learning methods have been developed for web crawlers that are capable to learn ontologies based on the information available online [4]. At the same time there is a lack of such mechanisms for agents working in other environments.

Additionally to the main part of the ontology – the concept hierarchy, it is necessary to have knowledge about the classes described in the hierarchy. Two approaches are used to represent knowledge like this. One option is to use mechanisms provided by the ontology description languages like OWL [9] to store additional rules about the actions and classes describing objects in the environment. Another option is to use additional custom knowledge base to store rules representing the additional knowledge about the classes. The second approach is more flexible, because richer knowledge representation mechanisms compared to the ones available in OWL specification can be used.

To enable full flexibility of the system, the following approach is preferable to creating the whole ontology before the deployment of the system. The main part of the ontology is created by human designer or by using some ontology learning technique. The major part of knowledge about these

concepts also is created by the designer of the system and inserted into the knowledge base. Afterwards the system must be capable to extend the knowledge based on the particular environment and the system's experience in it. The majority of the changes will be done in the knowledge base, but also the ontology must be extendable.

The aim of the research done at the Department of Artificial Intelligence and System Engineering is to create a framework that would enable the agents to collectively change the knowledge structure used in the multi-agent system and to actually implement the proposed method in one of the agent development platforms, in particular JADE platform [1]. This paper outlines the first step towards such a solution, namely, the conceptual framework that will be implemented as a part of the future research.

The remainder of the paper is organized as follows. Section 2 outlines the use case that has been applied for requirements definition and also later on will be used for validation of the proposed framework. Section 3 outlines the chosen knowledge representation approach. Section 4 proposes a knowledge update approach, including the machine learning method enabling the agents to learn new relationships between the objects and agent's actions based on the agents' experience. Section 5 concludes the paper and outlines the most important directions of future work.

2 Use case

Domains where autonomous agents need to find the most appropriate capability for the particular task usually require rich knowledge about the capabilities and actions to find the best match between agent's capabilities and tasks or actions that must be done in the environment (from now on in this paper will be named actions). This knowledge must include the semantics of both actions and capabilities. One of such domains is a fleet of heterogeneous cleaning robots working in indoors environment and cleaning different surfaces. The first multi-robot systems in the cleaning domain start to appear, for example [11], but there are no mechanisms in such systems to change the knowledge structure of the system after it is deployed. It may become a serious obstacle in creating fully autonomous multi-robot systems. Due to this reason this domain is chosen as a case study for the knowledge representation and update mechanism research.

The following model of the multi-robot system is considered. The environment consists of various surfaces that need to be cleaned. Two types of surfaces are considered in this paper, namely, floors and furniture. Further on, each type of the surfaces require different cleaning methods, in particular, some methods are more efficient in cleaning particular surface, some are useless, and some are even damaging the surface and thus are not allowed to be used. On the other hand each robot has particular capabilities or cleaning methods that it is equipped with. For example, a cleaning robot may have a vacuum cleaning capability and a wet cleaning capability. The aim of the decision making is to allocate the cleaning tasks to the most appropriate robots and as a consequence to get close to optimal cleaning result.

After creating a management system for task allocation among robots, both new objects for cleaning and new robots with new cleaning methods appear and as a consequence must be dealt with. Additionally, the types of the surfaces make a long (practically infinite) list and it is impossible to consider all of them before building the system. As a consequence the system must be capable to learn and adapt to new environments.

The study of this problem is done in a simulated environment where each robot is represented by an intelligent agent and the tasks are done in the software environment. The feedback about the quality of the results is evaluated by the user of the system (expert) according to the expert's knowledge about the applicability of the particular robot to the particular object.

3 Knowledge representation approach

The knowledge representation is done based on the ontology. Ontology is represented in the form supported by JADE framework, i.e., it is formed as Java class hierarchy. As a consequence all other knowledge except the class hierarchy is represented outside the ontology. The developed knowledge representation approach consists of the following parts:

- Class hierarchy, whose main parts are the class hierarchies of the capabilities and actions;
- Model of the environment stored in the form of ontology class instances;
- Set of rules defining the knowledge about the relations between capabilities and actions;
- Priority model for reasoning.

3.1 Class hierarchy

The class hierarchy defines the classes used to model the domain. It is stored in the form of Java classes using the JADE Bean Ontology [3]. The two

main hierarchies are included in the ontology. First, the hierarchy of actions represents the actions that the agents have to do in the environment. Second, the hierarchy of capabilities defines the capabilities that agents can have in this domain. Additionally any other classes that are used in the communications among agents can be defined in separate hierarchies.

In the use case considered in this paper and described in Section 2 the hierarchies contain the following classes. The first hierarchy is defining the areas to be cleaned. As discussed above, two types of areas are included in the case study, namely floor and furniture surfaces. As a consequence FloorCover and Furniture are the direct subclasses of the class Action. The second hierarchy consists of the cleaning methods that agents may be capable to do. The superclass of this hierarchy is CleaningMethod.

The Ontology Learning Tool for creating the knowledge structures for multi-agent systems has been developed. The classes of the ontology can be defined in this tool. The Java code of the classes can also be generated by the tool. Figure 1 contains the class hierarchy of the cleaning methods.

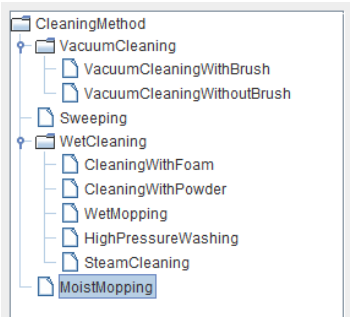

Fig. 1. Class hierarchy of cleaning methods

3.2 The environment model

The environment model is defined as a set of instances of the action classes defined in the ontology. These instances define the objects that the environment consists of. In the cleaning use case two types of instances are considered, namely, floor areas and pieces of furniture. Each instance describes a particular object that needs cleaning. For each of them the type of the object is defined by its class. Load of the object that defines how quickly the object gets dirty, importance and size of the object are specified as attributes. Additional attributes like location, accessibility, etc., can also be specified for a particular class of objects. The instances are stored in a centralised repository and are available to all agents during the operation of the system. User can create requests and give feedback to the system by stating facts about the particular instances and results of their cleaning.

3.3 The set of rules

The knowledge about the applicability of each capability is stored in the form of rules. These rules are used for the system to find the most appropriate capability for each action that the system has to do. In particular, in our use case for each time when some object must be cleaned a robot equipped with a suitable cleaning mechanism must be chosen.

The following types of rules are used in the proposed approach:

- The default rules defining relationships between actions and capabilities;
- Additional rules stating additional knowledge about particular classes or instances and their relationships.

The *default rules* are generated based on the list of cleaning methods specified by the designer during the definition of the furniture or floor cover class. These rules just state what capabilities (cleaning methods in the use case) are applicable to what actions (furniture and floor areas in the use case).

The *additional (user defined) rules* are explicitly defined by the user of the tool (designer of the particular multi-agent system). These rules are domain specific and can state specific facts about instances and/or classes. In the use case rules can state the following facts:

- A specific relationship among classes or between a class and an instance stating one of the following:
  - Always rule: stating that the cleaning method is the only one that should always be used for cleaning particular class or instance;
  - Never rule: stating that the cleaning method should never be used for cleaning particular class or instance;
  - Priority rule: increasing or decreasing the priority of the relationship among classes;
- Parameter rule: stating the appropriateness of a cleaning method for the particular values of attributes of the particular class of objects.

3.4 The priority model

The developed knowledge representation approach enables the use of a priority based decision making to find the most appropriate agent to carry out the particular action. In our use case it means to find the most appropriate agent and a cleaning method that it is equipped with for cleaning a particular object. The priority model is based on the following

assumptions. The values of priority are in a scale from 0 (lowest) to 100 (highest). The priority is found separately for each cleaning method that the agent has and the priority of the agent as a whole is the maximum value of its capabilities. The rules are used to find the priorities of cleaning methods in the following way:

- If there is never rule about the pair of cleaning method and object, then the priority is 0;
- If there is always rule about the pair of cleaning method and object, then the priority is 100;
- If there is none of the above rules and there is a default rule then the priority is 60;
- If there is none of the above rules and there is a default rule connecting superclass of cleaning method to the object class or vice versa, then the priority is 60-20*difference in hierarchy levels to the superclass. If the calculated priority is below 10, it is set to exactly 10.
- If there is a priority rule relating the corresponding instances, classes or superclasses then the corresponding increase or decrease of priority is done. If the calculated priority is below 10 or above 100, it is set to exactly 10 or 100 respectively.
- If there are no rules about the particular classes and instances then the priority is set to the default value of 30.
- If there is a parameter rule relating the corresponding instance, class or superclass to the particular value of an attribute then the corresponding increase or decrease of priority is done. If the calculated priority is below 10 or above 100, it is set to exactly 10 or 100 respectively.
- The priority is updated based on the estimated cleaning time including the time needed for the existing tasks of the particular agent. The priority is decreased by 5 for each time unit multiplied with the importance of the particular object (it has scale 1 to 5). A particular value of a time unit is defined based on the particular application and length of task execution.

3.5 The agents

The multi-agent system consists of one manager agent and a number of cleaning agents that simulate cleaning robots in the use case described above. The manager agent monitors the environment and whenever it identifies the need to clean a particular object it starts a Contract Net protocol [7] to allocate the task to the most appropriate cleaning agent. The agents use the priority model to calculate their appropriateness to the particular task and make bids equal to their priority values. The agent with the highest priority value is chosen to execute the task.

4 Knowledge structure update

The Section 3 outlined the knowledge representation approach. This section describes the architecture of the system that creates, uses and updates the described knowledge and knowledge structures. The architecture of the whole system is described first, followed by the descriptions of the particular components and concluded by the learning mechanism.

4.1 Architecture of the system

The architecture of the whole system is composed of the following main components: knowledge base, manager agent with its interfaces and the set of agents (see Figure 2). The knowledge base consists of the rule base, environment model that are both based on the class hierarchies defined in the ontologies, i.e., they are using classes from these hierarchies to define their instances and additional knowledge about them in the form of rules.

The relationships among the mentioned higher level components are the following. The agents use the knowledge base for their priority based decision making. They also use the ontology to define the semantics of the communications according to the JADE Ontology support [2]. Finally, agents collect the example set for the learning mechanism. The manager agent updates the knowledge base and ensures the fulfilment of user's requests by contracting the appropriate agent to do the particular task.

The remaining components or parts of the system, namely the multi-agent system, the manager agent as well as the rule based machine learning approach are described in details in the following subsections.

4.2 The multi-agent system

The multi-agent system (MAS) consists of two types of agents. The manager agent represents the user and acts on behalf of him/her to find the most appropriate performer of each task. A Contract Net protocol is used to allocate tasks to the most appropriate agent. The set of cleaning agents is heterogeneous in the sense that each agent can have different capabilities. The agents simulate cleaning robots. Each cleaning agent knows its capabilities and can use the common knowledge base to calculate its appropriateness to the particular task and to choose the particular method to execute it.

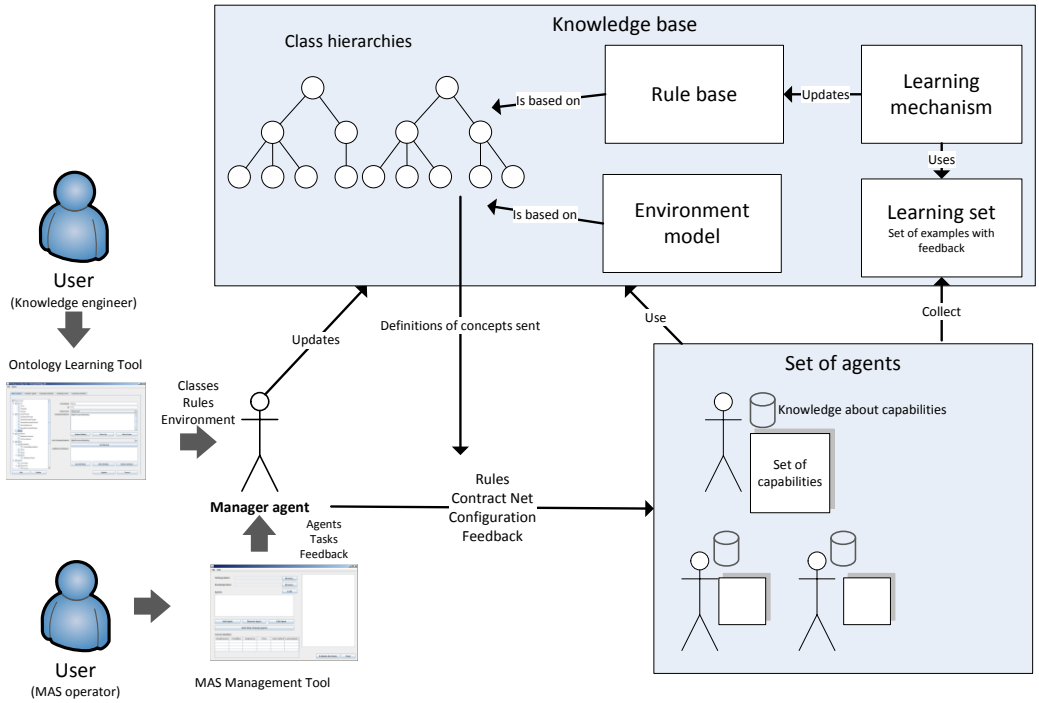

Fig. 2. The architecture of the system

The scenario for executing one cleaning task is the following:

1. The manager agent forms the task either based on explicit command by user or based on the environment monitoring. The need for cleaning in the virtual environment is simulated by a particular frequency of cleaning needed for each object.
2. The manager agent starts the Contract Net protocol and sends out the task description;
3. The cleaning agents calculate their priority values and make their bids accordingly;
4. The manager agent chooses the best agent from the bidders and awards the task;
5. The chosen cleaning agent does the job and reports when finished;
6. The user gives his/her evaluation about the cleaning result;
7. Based on the user's feedback the particular agent forms an example and adds it to the example set.

Steps 6 and 7 are done only during the learning period of the system. Later on they are omitted.

4.3 The manager agent

The manager agent's purpose is to serve as an interface between the system and its users and to organise the work of the set of other agents. It is an agent with two user interfaces, namely the user interface for knowledge engineer and the user interface for the operator of the system.

The user interface for the knowledge engineer is the Ontology Learning Tool, the functionality necessary for the following tasks:

- Defining the class hierarchies to be included in the ontology;
- Defining the environment model consisting of the class instances;
- Specifying the user defined rules.

Ontology classes according to the standards of JADE platform are generated by the tool based on the ontology defined in the tool, additionally the environment model and set of rules are saved as objects into a knowledge base file that can be later on used by the multi-agent system. Currently after generation of new ontology classes the system must be restarted and recompiled. Creation of a mechanism that allows changing the ontology during the runtime is a future work. The ontology can be also automatically extracted from an existing multi-agent system and improved in the Ontology Learning Tool.

The user interface for the end user or system's operator allows doing the following tasks:

- Configure the multi-agent system. The ontology, environment as well as the rule base can be specified. In the simulated environment it is possible to create the simulation scenario by specifying all the agents that will be part of the system and their capabilities.

- Make requests to do a particular task, i.e., to clean a particular object;
- To follow the simulation of the cleaning system in terms of tasks done by particular agents and their results;
- Giving feedback about the quality after the cleaning of a particular object is finished.

4.4 The Learning Mechanism

Not all facts needed for the operation of the multi-agent system can be known during the design time. Important characteristics can change even during the runtime of the system. For example, new types of objects (furniture or floor covers) can be introduced into the environment, new cleaning methods can be introduced into the working system or some other important characteristics of the system may change. Additionally, the knowledge of the designer of the system about the particular environment may be imprecise. All of these reasons lead to the situation where the system is not capable to make optimal decisions. To deal with such a situation an autonomous system that is capable to learn new details about the particular environment is needed. The following learning approach is proposed to increase the autonomy of multi-agent systems.

The system starts operating based on the set of rules defined by the knowledge engineer in the Ontology Learning Tool and stored in the rule base. After completing every task the user may give a feedback about the result of the cleaning. In the simulated environment the user must base his/her opinion on theoretical knowledge about the combination of the object and the cleaning method used while in real environment the feedback can be given based both on the experts knowledge and practical evaluation of the result. The feedback is collected in the example set in the traditional form for the inductive learning, namely the values of the attributes (cleaning method, object, and class of the object) and the evaluation. The evaluation is provided in the scale from (-3 as very bad to +3 as the best possible choice, 0 is used in case no feedback is available).

The example set is used to calculate the offsets of priorities for the particular combinations. Offsets are recalculated after collecting a particular number of examples. It is done by adding up the evaluation of the following pairs and getting the corresponding offsets:

- The particular cleaned object's instance and the cleaning method of the particular agent;
- The cleaned object's class and the cleaning method of the particular agent;
- The particular cleaned object's instance and the cleaning method's class;
- The cleaned objects class and the cleaning method's class.

The resulting offset values for each pair are stored in the knowledge base together with the corresponding rules. These values are used during each calculation of the appropriateness by adding up all the offsets that are related to the particular situation, i.e., the offset corresponding to the cleaning method class and object class, cleaning method class and particular object (instance), particular cleaning method and object class as well as particular object and particular cleaning method. As a result, if exactly the same combination of a cleaning method and object's instance is evaluated later on the offset is maximal while the offset for similar combinations (i.e., the objects of the same class) then the offset is smaller. The calculated offset is multiplied to the learning rate (see Equation 1) and added to the priority calculated based on the rules given in the Section 4.2.

$$O = \alpha * \sum_{i,j} O_{i,j}, \quad \text{where} \qquad (1)$$

O – the final offset;
α – learning rate;
$O_{i,j}$ – offset of a particular pair that is related to the particular combination of object and cleaning method.

In the particular case study learning rate is set to 1, but in other cases the author sees that lower values can also be used for more conservative learning speeds.

Additionally to the described learning method new classes can be added to the class hierarchies by using Ontology Learning Tool. Whenever it is done, new ontology files are generated and the corresponding files are replaced in the system. Similarly, also the existing set of rules can be edited. Whenever any rule is added, removed or changed all the offsets related to the particular pairs of cleaning methods and objects are removed and the learning is started again from scratch. It is done to prevent combining new rules with the offsets calculated based on previous version of rules that can be misleading together with the new versions of the rules.

5 Conclusions

An approach for adaptive knowledge representation in multi-agent systems is proposed. It combines

knowledge representation, knowledge structure updating and learning mechanisms into the same framework. An Ontology Learning Tool is developed for defining and editing ontologies as well as for generating ontology classes to be used in multi-agent systems. Additionally, the tool has functionality for defining the contents of the knowledge base in the form of environment definition and rules. The rules defined by the designer serve as background knowledge for learning algorithm that finds the more precise preferences between actions in particular situations based on the system's experience. The proposed approach enables development of multi-agent systems in domains with only limited understandings about priorities between the capabilities of different agents. The job of finding the exact priorities is done by the learning algorithm whose task is to do the fine-tuning of the values. Such a combination is preferred to only user defined rules or only machine learning approach because the user defined rules allow learning the exact representations relatively rapidly. The learning only approach requires significantly more trials to learn the same result while the user defined rules only limit the flexibility of the system.

During this paper the proposed approach is implemented and explained on the example of cleaning agents, but the approach is general enough and can be applied to any domain where the system must find the most appropriate capabilities of the group of agents for doing different tasks.

The main direction of the future work is to finish the implementation of the simulation environment for testing purposes of the proposed concept. It will enable practical validation of the proposed approach.

Acknowledgement

This work is partly supported by the Latvian National research program SOPHIS under grant agreement Nr.10-4/VPP-4/11.

References:

[1] Bellifemine F.L., Caire G., Greenwood, D. *Developing Multi-agent Systems with JADE*, Wiley Series in Agent Technology, 2007, 300p.

[2] Caire, G., Cabanillas, D., *Jade Tutorial Application-Defined Content Languages and Ontologies*, Technical Report, Telecom Italia, 30 p., 2010 Available online: http://jade.tilab.com/doc/tutorials/CLOntoSupport.pdf (last visited 18.02.2016).

[3] Cancedda, P., Caire., G. *JADE Tutorial Creating Ontologies by Means of the Bean-Ontology Class*. Telecom Italia. 2010. Available Online: http://jade.tilab.com/doc/tutorials/BeanOntologyTutorial.pdf (Last Visited 04.03.2016).

[4] Dong, H. and Hussain, F. K. *SOF: a semi-supervised ontology-learning-based focused crawler*. Concurrency and Computation: Practice and Experience, 25, 2013, pp. 1755–1770.

[5] Drumond L., Girardi, R., *A Survey of Ontology Learning Procedures*, Proceedings of the 3rd Workshop on Ontologies and Their Applications collocated with SBIA 2008, Salvador, Brazil.

[6] *FIPA ACL Message Structure Specification*, 2002. Available Online: http://www.fipa.org/specs/fipa00061/SC00061G.pdf (Last visited 18.02.2016).

[7] *FIPA Contract Net Interaction Protocol Specification*, Geneva, Switzerland, 2002, Available Online: http://www.fipa.org/specs/fipa00029/SC00029H.pdf (Last visited 01.03.2016).

[8] Hazman, M., El-Beltagy, S.R., Rafea, A., *A Survey of Ontology Learning Approaches*. International Journal of Computer Applications, Volume 22–No.9, May 2011, pp 36-43.

[9] Heflin J., *An Introduction to the OWL Web Ontology Language*. Available Online: http://www.cse.lehigh.edu/~heflin/IntroToOWL.pdf (Last visited 25.02.2016).

[10] Lehmann, J., Voelker, J. *An Introduction to Ontology Learning*, in Jens Lehmann & Johanna Voelker, ed., 'Perspectives on Ontology Learning' , AKA / IOS Press, 2014, pp. ix-xvi .

[11] Nikitenko, A., Grundspenkis, J., Liekna, A., Ekmanis, M., Kulikovskis, G., Andersone, I. *Multi-Robot System for Vacuum Cleaning Domain*. Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection: Proceedings of the 12th International Conference (PAAMS 2014), Spain, Salamanca, 4-6 June, 2014. Heidelberg: Springer, 2014, pp.363-366.

[12] Zouaq A.,Gasevic, D., Hatala, M., *Unresolved Issues in Ontology Learning* - Position Paper - Proceedings of CSWS2011, 2011, pp. 52-57.