Approach to translation of RDF/OWL-ontology to the graph knowledge base of intelligent systems

A Filippov[1], V Moshkin[1], A Namestnikov[1], G Guskov[1] and M Samokhvalov[1]

[1] Ulyanovsk State Technical University, Ulyanovsk, Russia

Abstract. The paper presents the graph knowledge base of the ontology repository. The original algorithm for translating the RDF/OWL-ontology into a graph knowledge base is proposed. In addition, the article presents an approach to the inference on the ontology repository. The approach based on translating the SWRL constructs into the elements of the Cypher language. Examples of using the proposed algorithms and models are presented in the article.

1. Introduction

Post-industrial society operates with huge volumes of information both in everyday and professional activities. A large amount of information causes difficulties in making decisions within the framework of tight time constraints [1] [2].

A variety of software automation of human activities are used to solve this problem. However, it is necessary to adapt them to the specifics of a particular problem area (PrA).

Ontologies are usually used to describe the subject area features [3] [4] [5] [6] [7] [8] [9]. Ontologies could be presented as subject area models in the form of a semantic web (graph) [10] [11].

The RDF and OWL languages are the most common formats to represent ontologies [12] [13] at the current time. The OWL language is expressive and contains various kinds of functional relationships.

The Protégé editor is the primary and most commonly used tool for building ontologies. The user must have knowledge in the ontological design and knowledge engineering fields for working with Protege.

The OWL API library is often used to work with ontologies during the operation of intelligent systems [14]. The OWL API has the greatest functionality [15], but it can only be used in programs written for the JVM platform (Java Virtual Machine).

The SPARQL query language [16] is usually used to generate queries for the RDF-ontologies. The reasoners (Pellet [17], Fact++ [18], Hermit [19]) are used for the OWL-ontologies.

Thus, there is no unified method of working with ontologies and generating queries to them.

One solution of this problem is specialized repositories, for example, StarDog [20], Virtuoso [21], RDF4j [22], etc. However, existing ontology repositories have the following disadvantages:

- need to acquire licenses for use;
- only RDF format full support.

Our scientific group has developed a new efficient ontology repository. This repository:

1. Imports RDF and OWL ontologies.
2. Creates queries to the repository.
3. Does not require the developer knowledge in the ontological design and knowledge engineering.

4. Interacts with the ontology repository using the HTTP protocol.
5. Makes the repository independent of the programming language and operating system used.

2. The model of the KB of the ontology repository

Ontology is a model of the PrA representation and visualized in the form of a semantic graph.

Graph-oriented database management system (Graph DBMS) Neo4j [23] is the basis of the ontology store for fuzzy KB. Neo4j is currently one of the most popular graph databases and has the following advantages:

1. Having a free community version.
2. Native format for data storage.
3. One copy of Neo4j can work with graphs containing billions of nodes and relationships.
4. The presence of a graph-oriented query language Cypher.
5. Availability of transaction support.

Neo4j was chosen to store the description of the PrA in the applied ontology form since the ontology is actually a graph. In this case, it is only necessary to limit the set of nodes and graph relations into which ontologies on RDF and OWL will be translated.

The context of an ontology is some state of ontology obtained during versioning. Context can also be a subject area.

Formally the ontology is:

$$O = \left\langle T, C^{T_i}, I^{T_i}, P^{T_i}, S^{T_i}, F^{T_i}, R^{T_i} \right\rangle, i = \overline{1,t}, \quad (1)$$

where t is a number of the ontology contexts,

$T = \{T_1, T_2, \text{K}, T_n\}$ is a set of ontology contexts,

$C^{T_i} = \left\{C_1^{T_i}, C_2^{T_i}, \text{K}, C_n^{T_i}\right\}$ is a set of ontology classes within the i-th context,

$I^{T_i} = \left\{I_1^{T_i}, I_2^{T_i}, \text{K}, I_n^{T_i}\right\}$ is a set of ontology objects within the i-th context,

$P^{T_i} = \left\{P_1^{T_i}, P_2^{T_i}, \text{K}, P_n^{T_i}\right\}$ is a set of ontology classes properties within the i-th context,

$S^{T_i} = \left\{S_1^{T_i}, S_2^{T_i}, \text{K}, S_n^{T_i}\right\}$ is a set of ontology objects states within the i-th context,

$F^{T_i} = \left\{F_1^{T_i}, F_2^{T_i}, \text{K}, F_n^{T_i}\right\}$ is a set of the logical rules fixed in the ontology within the i-th context,

R^{T_i} – is a set of ontology relations within the i-th context defined as:

$$R^{T_i} = \left\{R_C^{T_i}, R_I^{T_i}, R_P^{T_i}, R_S^{T_i}, R_F^{T_i}\right\}, \quad (2)$$

where

$R_C^{T_i}$ is a set of relations defining hierarchy of ontology classes within the i-th context,

$R_I^{T_i}$ is a set of relations defining the 'class-object' ontology tie within the i-th context,

$R_P^{T_i}$ is a set of relations defining the 'class-class property' ontology tie within the i-th context,

$R_S^{T_i}$ is a set of relations defining the 'object-object state' ontology tie within the i-th context,

$R_F^{T_i}$ is a set of relations generated on the basis of logical ontology rules in the context of i-th context.

Some relations of the G ($R_C^{T_i}$ и $R_I^{T_i}$) may be functional relations. Functional relations are characteristic of the OWL language.

3. Translation of RDF/OWL-ontology into a graph KB

It is necessary to select structural elements of TBox (structure, scheme) and ABox (content) of graph KB respectively for successful translation of RDF or OWL ontology to graph KB objects.

Formally the functions of translating an RDF/OWL ontology to a graph KB are:

$$f_O^{RDF} : RDF \to O,$$

$$f_O^{OWL} : OWL \to O,$$

where $RDF = \left\langle C^{RDF}, I^{RDF}, P^{RDF}, S^{RDF}, R^{RDF} \right\rangle$ – set of entities RDF ontology (corresponds to the entities of expression 1),

$OWL = \left\langle C^{OWL}, I^{OWL}, P^{OWL}, S^{OWL}, R^{OWL} \right\rangle$ – set of entities RDF ontology (corresponds to the entities of expression 1),

O – set of ontology entities of the graph KB (expression 1). Table 1 shows the correspondence of the RDF / OWL-ontology entities to the graph KB entities.

Table 1. Correspondence of RDF / OWL-ontology entities to the graph KB entities

RDF	OWL	Graph KB
	TBox	
rdfs:Resource	owl:Thing	$C^{T_i} = \left\{C_1^{T_i}, C_2^{T_i}, \text{K}, C_n^{T_i}\right\}$
rdfs:Class	owl:Class	$C^{T_i} = \left\{C_1^{T_i}, C_2^{T_i}, \text{K}, C_n^{T_i}\right\}$
rdfs:subClassOf	owl:SubclasOf	$R_C^{T_i}$
rdf:Property	owl:ObjectProperty owl:DataProperty	$P^{T_i} = \left\{P_1^{T_i}, P_2^{T_i}, \text{K}, P_n^{T_i}\right\}$
rdfs:domain	owl:ObjectPropertyDomain owl:DataPropertyDomain	$R_P^{T_i}$
rdfs:range	owl:ObjectPropertyRange owl:DataPropertyRange	$R_P^{T_i}$
	ABox	
rdf:type rdf:ID	owl:NamedIndividual owl:ClassAssertion	$I^{T_i} = \left\{I_1^{T_i}, I_2^{T_i}, \text{K}, I_n^{T_i}\right\}$ $R_I^{T_i}$
rdf:resource rdf:ID	owl:ObjectPropertyAssertion owl:DataPropertyAssertion	$S^{T_i} = \left\{S_1^{T_i}, S_2^{T_i}, \text{K}, S_n^{T_i}\right\}$ $R_S^{T_i}$

The main entities of RDF and OWL ontologies correspond to the ontology of the graph KB.

The graph KB entities unify different formats of ontologies and form a data model. The developer can build queries to the ontology repository contents on Cypher using the generated data model. This method of extracting knowledge from the ontology repository is more familiar to the developer than working with reasoners. However, the possibility of the inference on the contents of our repository of ontologies exists.

4. The inference on the KB contents

The inference is the process of reasoning from the premises to the conclusion. Reasoners are used to implement the function of inference. Reasoners form logical consequences produce an inference based on many statements, facts and axioms [24] [25] and also carry out a control of logical integrity.

The Neo4j GDBMS does not allow the use of existing reasoners. Thus, there is a need to develop a mechanism for inference from the ontology repository content.

Formally the logical rule of the ontology of the fuzzy knowledge base is:

$F^{T_i} = \left\langle A^{Tree}, A^{SWRL}, A^{Cypher} \right\rangle$,

where

T_i – i-th context of the ontology of the fuzzy KB;

A^{Tree} – a tree-like representation of a logical rule F^{T_i};

A^{SWRL} – SWRL representation of the logical rule F^{T_i};

A^{Cypher} – Cypher representation of the logical rule F^{T_i}.

The tree-view A^{Tree} of a logical rule F^{T_i} is:

$A^{Tree} = \langle Ant, Cons \rangle,$

where

$Ant = Ant_1 \Theta Ant_2 \Theta \text{K } Ant_n$ – is the antecedent (condition) of the logical rule F^{T_i};

$\Theta \in \{AND, OR\}$ – is a set of permissible logical operations between antecedent atoms;

$Cons$ – consequent (consequence) of a logical rule F^{T_i}.

Figure 1 shows an example of a tree-like representation of a logical rule. This rule describes the nephew-uncle relationship.

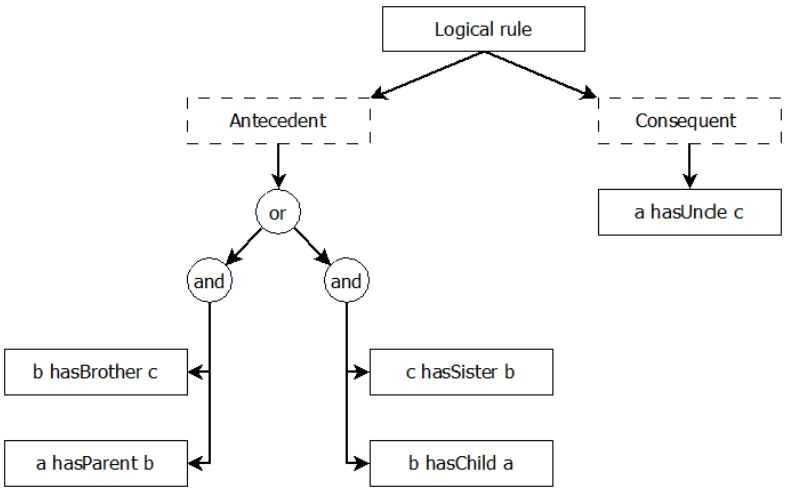

Figure 1. Example of a tree-like representation of a logical rule.

The tree-like logical rule is translated into the following SWRL [26]:

1. *hasParent(?a,?b) & hasBrother(?b,?c) & => hasUncle(?a,?c)*
2. *hasChild(?b,?a) & hasSister(?c,?b) => hasUncle(?a,?c),*

and the following Cypher view:

1. MATCH (s1:Statement{name: "hasChild", lr: true})
MATCH (r1a)<-[:Domain]-(:Statement{name:"hasFather"})-[:Range]->(r1b)
MERGE (r1b)-[:Domain]->(s1)
MERGE (r1a)-[:Range]->(s1)
2. MATCH (s1:Statement{name: "hasChild", lr: true})
MATCH (r2c)<-[:Domain]-(:Statement{name:"hasSister"})-[:Range]->(r2a)
MATCH (r2c)<-[:Domain]-(:Statement{name:"hasFather"})-[:Range]->(r2b)
MERGE (r2b)-[:Domain]->(s1)
MERGE (r2a)-[:Range]->(s1).

Thus, the rules are translated into their tree-view when imported into the KB of logical rules in the SWRL language.

The presence of a tree-like representation of a logical rule allows you to form both a SWRL-representation of a logical rule and a Cypher-representation based on it.

Relations of a special type - $R_F^{T_i}$ (expression 2) are formed by using a logical rule on the Cypher between the graph KB entities F^{T_i} (expression 1). The graph KB entities must satisfy the atoms of the antecedent of the logical rule. Relations formed to organize the inference from the ontology repository contents.

A set of Cypher queries is formed to control the logical integrity of the ontology repository contents. Cypher queries invert the axioms of TBox. If one of these queries returns the result then the logical integrity is violated.

5. Conclusion

This article considers the approach to translating RDF/OWL-ontology in the graph KB. The developed ontology repository based on the graph KB allows the developer to interact with the repository contents in the most familiar way - to build queries to the graph KB content.

In this case, the unified graph DB data model (expressions 1 and 2) allows you to import RDF and OWL ontologies.

These formats are often used to form a description of the domain features in the ontology form. TBox of graph KB (Table 1) defines the axioms available for formation and allows you to control the logical integrity of the graph knowledge base content.

The inference engine built into our ontology repository allows you to import SWRL rules and create a tree-like view of the SWRL rules based on them. The tree view allows you to create a set of logical rules in the graph KB. The rules of the graph KB can be used to derive new facts and axioms based on the graph KB contents.

Thus, the developer will need less time to study the ways and means of working with ontologies in the development of intelligent systems if he uses our ontology repository as a KB. The saved time will be used to develop business logic.

6. References

[1] Bova V., Kureichik V., Nuzhnov E. Problems of representation of knowledge in integrated systems of support of managerial decisions // Izvestiya SFU. Vol. 108 (7) (2010).

[2] Chernyakhovskaya L., Fedorova N., Nizamutdinova R. Intelligent support of decision-making in the operational management of business processes of the enterprise // Bulletin of the USATU. Management, Computer Science and Informatics. T 15. Vol. 42 (2) (2011).

[3] Vagin V., Mikhailov I. Development of the method of integration of information systems based on metamodelling and ontology of the subject domain // Software products and systems. № 1 (2008).

[4] Gavrilova T. Ontological approach to knowledge management in the development of corporate information systems // News of Artificial Intelligence. № 2 (2003).

[5] Zagorulko Yu. Construction of portals of scientific knowledge on the basis of ontology // Computational technologies. Vol. 12. No. S2 (2007).

[6] Karabach A. Information integration systems based on semantic technologies // Science, technology and education. Vol. 2 (2). (2014).

[7] Smirnov S. Ontological modeling in situational management // Ontology of designing. № 2 (2012).

[8] Tuzovsky A. Development of knowledge management systems based on a single ontological knowledge base. // Bulletin of the Tomsk Polytechnic University. 2007. T. 310. Vol. 2.

[9] Filippov A., Moshkin V., Shalaev D., Yarushkina N. Single ontological platform for data mining // Proceedings of the VI International Scientific and Technical Conference OSTIS-2016. Minsk. Republic of Belarus (2016).

[10] Moshkin V., Yarushkina N. Methods for constructing non-clear ontologies of complex subject domains // Proceed. of fifth international. scientific-techn. conf. OSTIS-2015. Minsk. pp. 401-406 (2015).

[11] Tarasov V., Kalutskaya A., Svyatkina M. Granular, fuzzy and linguistic ontologies to provide mutual understanding between cognitive agents // Proceed. of second international. scientific-techn. conf. «OSTIS-2012». Minsk. pp. 267-278 (2012).

[12] Resource Description Framework (RDF), https://www.w3.org/RDF, last accessed 2018/05/11.

[13] OWL Web Ontology Language Overview, https://www.w3.org/TR/owl-features, last accessed 2018/05/11.

[14] The OWL API, http://owlcs.github.io/owlapi, last accessed 2018/05/11.

[15] Owlcpp: a C++ library for working with OWL ontologies, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4574266, last accessed 2018/05/11.

[16] SPARQL Query Language for RDF, https://www.w3.org/TR/rdf-sparql-query, last accessed 2018/05/11.

[17] Pellet, http://pellet.owldl.com, last accessed 2018/05/11.

[18] FaCT++, http://owl.man.ac.uk/factplusplus, last accessed 2018/05/11.
[19] HermiT OWL Reasoner. The New Kid on the OWL Block, http://www.hermit-reasoner.com, last accessed 2018/05/11.
[20] The Knowledge Graph Platform for the Enterprise, https://www.stardog.com, last accessed 2018/05/11.
[21] About OpenLink Virtuoso, https://virtuoso.openlinksw.com, last accessed 2018/05/11.
[22] Eclipse RDF4J, http://rdf4j.org, last accessed 2018/05/11.
[23] Introducing the Neo4j Graph Platform, https://neo4j.com, last accessed 2018/05/11.
[24] Makhortov S. On the Algebraic Model of a Distributed Production System. // Proceedings of the Fifteenth National Conference on Artificial Intelligence with International Participation "KII-2016". Smolensk. Vol.1. pp. 64-72 (2016).
[25] Moshkin V., Yarushkina N. The inference based on fuzzy ontologies // Integrated models and soft computations in artificial intelligence. Proceedings of the VIII International Scientific and Practical Conference (Kolomna, May 18-20, 2015). Vol.1. Moscow. Fizmatlit. pp. 259-267 (2015).
[26] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, https://www.w3.org/Submission/SWRL, last accessed 2018/05/11.

Acknowledgments
This study was supported Ministry of Education and Science of Russia in framework of project № 2.4760.2017/8.9 and by the Russian Foundation for Basic Research (Grants No. 18-47-730035 and 16-47-732054).