Verifying Description Logic Ontologies based on Competency Questions and Unit Testing

Camila Bezerra[1,2], **Fred Freitas**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco
(UFPE) – Recife - PE – Brazil

{cbs,fred}@cin.ufpe.br

camila.bezerra.br@gmail.com

Abstract. *Ontology testing can help guarantee ontology quality, by finding errors and inconsistencies in the ontology. There are few approaches for ontology testing inspired in methods employed by software engineering, and usually they depart from software requirements. Requirements in ontologies are the set of competency questions. A drawback of some ontology testing approaches resides on the fact that testing takes place only at the assertional level, since the query language SPARQL is relied on. In this work, we propose an approach and its implementation to test semi-automatically OWL-DL ontologies with both assertional and terminological queries, by using competency questions and the concept of unit testing. The tests accomplished in small ontologies sounds promising with good results.*

1. Introduction

With the growing interest in ontologies, several methodologies have appeared to build them, some with similar aspects to methodologies employed in software engineering [Uschold and Gruninger 1996][Fernández-López et al. 1997][Suárez-Figueroa et al. 2012]. In ontology development, ontology verification means that the ontology was building correctly, according to expected [Gómez-Pérez 2004].

As an approach to enable ontology verification, ontology testing is a way to detect errors and inconsistences during (or in the end) of development phase. It enables the development of an ontology with a minimal quality, i.e. consistency and completeness. However, there are few tools and methodologies for ontology testing [Blomqvist et al. 2012] [Vrandečić and Gangemi 2006]. In a scenario with OWL-DL ontologies, none of them supports tests the terminological level.

Current approaches to test OWL ontologies work only at the assertional level, by using the query language SPARQL [Angles and Gutierrez 2008]. SPARQL is a query language that is as expressive as relational algebra [Angles and Gutierrez 2008] which yelds a lack of expressive power and reasoning capabilities to check OWL ontologies. In other words, SPARQL is not shaped to entail an answer which may be deduced by the ontology using subsumption if not made explicit in the ontology.

Competency questions (CQs) can play a crucial role in this cenario and they consist of a set of questions stated and replied in natural language, that the ontology must be able to answer correctly [Noy and Hafner 1997]. They play an important role both in requirements specification as well as in the evaluation phase of most ontology engineering methodologies.

Bearing this in mind, in this work we propose an approach and a tool to test semi-automatically OWL-DL ontologies with both ABox and TBox queries, by using competency questions and checking whether they are being met or not. We rely on the idea of unit testing, which aims to testing the individual units of a code in Software Engineering [Pressman 2001].

This paper is organized as follows: section 2 describes presents the proposed method; section 3 presents a tool that implements the method, section 4 shows preliminary results, section 5 is devoted to related work; and section 6 presents some conclusions and future work.

2. A method for ontology testing

To introduce our ontology testing approach with DL, we start by presenting an ontology formalization:

Definition 1. ***Vocabulary of Ontology***. *Voc(O) is the vocabulary, i.e., defined and used in ontology O, defined over the triple (N_C, N_R, N_O) as explained before. We denote by O $\models \delta$ if δ is true in all the possible models of O.*

Definition 2. ***Competency Questions***. *A competency question is a $\langle Q, \sigma \rangle$ such that Q is a query expressed in a formal language and σ is an answer to this query expressed as a variable substitution.*

Definition 3. ***Satisfied competency question***. *A competency question $\langle Q, \sigma \rangle$ is satisfied by an ontology O if $O \cup Q \models \sigma$.*

Since we are dealing with controlled natural language, we are dealing with some patterns of CQs that we have found, below some of them:

Pattern: Which + <class> + <property> + <class> + not + <property> + <class>?
Pattern: From which + <property> + <class>?
Pattern: Is + <individual>+ a +<class>?
Pattern: Is <class> + <individual> + or + <individual>?

In Software Engineering, the phase of testing is very important to guarantee software quality, since it aims at detecting errors, and, to check if the software was implemented in accordance with the requirements and user expectations. The motivation of this work is applying the very same idea in the context of Ontology Engineering.

Analogously, CQs represent user demands for knowledge regarding a domain of discourse. They usually enable developers to define the ontology. Given that domains of discourse and user requirements may change through time, ontologies should evolve in order to represent them accordingly. Thus, it is important that a permanent correspondence among the users goals and the ontology requirements exists. This must be accomplished using a method of systematic testing.

In this paper, we propose description logic ontology testing, relying on competency questions as the units to test single requirements. Following, we formalize the components of an ontology testing approach.

Definition 4. ***Test case, test plan***. *A Test case is represented by a CQ that is represented as already defined pair $\langle Q, \sigma \rangle$. A Test Plan contains a set of test cases, besides other information like author and date. A Test Plan is represented by $\{\langle Q, \sigma \rangle\}_{i=1}^{n}$,where n is the number of CQs.*

Definition 5. ***Test suite and execution result***. *A Test Suite is a subset of Test cases of a Test Plan, i.e, if ts is a Test Suite, and tp is its Test Plan then $ts \subseteq tp$. A test suite is important to reuse a set of test cases between test executions. A Test Execution result is an instance of a test suite that corresponds a set of test cases selected to be tested, the result of the tests, the author the performed the test, and date. The execution result is represented by the tuple $< \{\langle Q, \sigma \rangle_{i=1}^{n}\}, \{\langle Q, \sigma' \rangle_{i=1}^{n}\} author, date >$, where σ is the expected result, and σ' is the result of the test for each CQ of the suite, whereas the suite has n CQs.*

Definition 6. ***Base Ontology and Testing Ontology***. *Given an ontology O, we assume that exists (defined by the developer or by the test engineer) a small set of O that represents a base set Ob, so that $Ob \equiv \sum_{a \in setofaxiomsfromO} a \cup \sum_{i \in setofinstancesofO} i$,on the other hand, we assumed that the complement of Ob is Ot, which is the part the needs to be tested, such that $O = Ob \cup Ot$, e $Ob \cap Ot = \emptyset$.*

Definition 7. ***Representative Testing Ontology***. *Given an ontology O, a representative testing ontology Ot' is a subset of axioms and/or a set of representative instances from Ot, defined by the test engineer as $Ot' \equiv \sum_{a \in setofaxioms} a \cup \sum_{i \in setofaxioms} i$.*

These definitions rely in the initial knowledge contained in the ontology, i.e. the base ontology, which requires no competency questions. An example of such type of ontology can be upper-domain ones, like BTL2 [Schulz and Boeker 2013] and GFO-Bio [Hoehndorf et al. 2008]. The extending ontology is the one required to be tested. Therefore, not the whole content, e.g. classes and individuals, but a representative set selected by the ontology engineer should and can be used for testing.

3. OWL TESTING TOOL

With the aid of this tool, a user can manage test plans, test suites, test cases and execute test suites through the modules of the layer. First, it is required to create a test plan that contains test cases. As put before, we consider here a test case as a CQ. Figure 1 displays our testing method steps. Considering the Pizza ontology, a test case could be “From which nation is the American Pizza”, “America”, where “From which nation is the American Pizza” is the small piece from the ontology to be tested, while “America” is the expected result from the test. In the test execution, besides the test case there is a field that indicates if the test case passes or not.

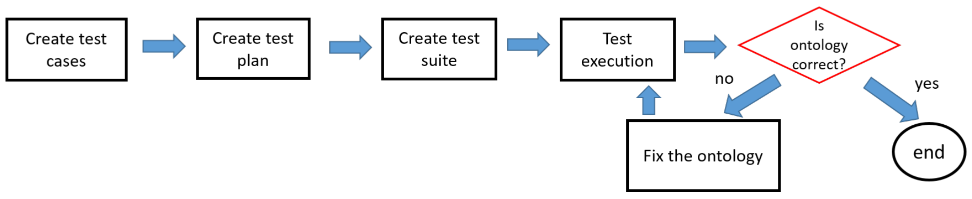

Figure 1. Cycle of testing ontology method

In a test execution module, the user chooses which test suite will be executed and the ontology to be tested. The execution is performed by the CQChecker, a tool to support the automation of verifying CQs against OWL ontologies. It provides a mechanism to verify whether the ontology meets its corresponding CQs. CQChecker supports both assertional and terminological queries.

CQChecker module [Bezerra et al. 2014] supports the automation of checking CQs during Ontology Evaluation, and particularly for asserting functional requirements expressed as CQs. The basic functioning of the tool can be summarized in the following terms: first, it analyzes the CQ in order to classify it into one of three types, according to the possible answer it is supposed to retrieve (over classes or instances). Then, the system directs the CQ to the corresponding module, where it will be converted and checked. To accomplish this, our algorithm [Bezerra et al. 2014] basically takes a CQ, splits it into tokens and tries to find the concepts and relations from the ontology described in OWL DL, which the CQ referred to.

Consider the CQ "What is the base of Real Italian Pizza?" about the Pizza ontology. With it and the CQChecker, we want to check if there is any class that would provide an individual for the image of RealItalianPizza, through the relation hasBase, which is a PizzaBase. For this to be achieved, we first search for an object property which can be a verb or a noun, and afterwards we look for the class, which has an image that is a PizzaBase in the relation, in this case ThinAndCrispyBase.

4. Preliminaries Experiments

As a validation, we performed tests with three ontologies: Pizza, Travel and Wine ontologies, which are available on the Protégé website. We created a test plan with 12 test cases and 3 test suites. The preliminaries experiments were preformed by the authors.

The CQs are related to several constructs of OWL-DL like class hierarchies, individuals, disjoint classes, intersection (A ∩ B) and union of classes (A ∪ B); equivalent classes($A \equiv B$), universal ($\forall$), existential quantification($\exists$) and "has-value" restrictions; and cardinality restrictions.

We performed a test execution in each of them. Each test suite contains the test result. The tool deploys the CQ set, so that the user can select CQs to test. The user can also compare the answer of the test with the expected answer, thus checking if the test passed.

The first measured used is accuracy that is % of execution test correctly answered by the tool. For the Pizza test we get 90% of accuracy. For the Travel test we get 80% of accuracy, and for the Wine test we get 90% of accuracy.

We show now a demonstration of our approach with the pizza ontology. First we create the test cases to the test plan and two test suites available in `https://www.dropbox.com/sh/ec6dwv83anqp64s/AAB-7VgJytljII9f5jqpgw2sa?dl=0`. The test suite number 1 is showed below:

CQ1: **What is the spiciness of a chicken Topping?** Correct answer: *Mild*

CQ2: **What is the base of Real Italian Pizza?** Correct answer: *ThinAndCrispyBase*

CQ3: **What is the country of Origin of American Pizza?** Correct answer: *American*

CQ4: **Which are the pizzas disjoint of Vegetarian Pizza?** Correct answer: *NonVegetarianPizza*

Considering an user named joseph that takes this test suite to do a test execution on 08/10/2016. The goal is to verify if the ontology follows this set of CQs.

So, Joseph load the pizza ontology, the test suite number 1, and test each CQ. The tool returned an answer that can be equal or not the correct answer. If the the returned answer is equal to correct answer then, Joseph will check this CQ as correct. In the end, the tool will save the results from the table, moreover who did and date, in this case, Joseph and the date 08/10/2016.

5. Related Work

Few engineering methodologies cite how, for what purpose, and by which means the ontology engineer has to use CQs. However, there are few proposals of test ontologies approaches. [Vrandečić and Gangemi 2006] discusses in their paper the need for unit testing and describes some possible approaches that can be applied. However, it seems to be a position paper, as no concrete approach or tools were provided.

[Blomqvist et al. 2012] provides a methodology and a tool for dealing with ontology testing. It resembles ours in some aspects; however, they use CQs only at the assertional level by relying on SPARQL. Also, a method and a tool called OntologyTest been proposed to test the functional requirement of an ontology-OWL DL by [García-Ramos et al. 2009]. The queries must be defined in SPARQL.

In relation to these works, ours deals at both assertional and terminological levels. Moreover, while [Blomqvist et al. 2012] represent the test plan and test cases as an ontology, our approach stores them in CSV files, to provide more flexibility to the ontology engineering.

6. Conclusions

In this paper, we proposed a new method to ontology testing based on competency questions for OWL-DL ontologies, for this, we use the concept of unit testing. Furthermore, we proposed a tool that implements this method. Ontology testing helps to guarantee the quality of ontologies, by detecting errors and inconsistences in the ontology. Competency questions can be used both in requirements specification as in evaluation phase.

There are some limitations of our approach, for instance, only treats simple English sentences by identifying key words. It is necessary to build a mechanism to accept complex sentences like “Does a bouquet or body of a specific wine change with vintage?”.

For future work, solving these deficiencies is certainly the main task. However, in general, we believe that our approach and the tool can make the job of to test description logic ontologies efficiently and rapidly. And, we intend in the future to make experiments with other users outside of the project.

References

Angles, R. and Gutierrez, C. (2008). The expressive power of sparql. In Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., and Thirunarayan, K., editors, *The*

Semantic Web - ISWC 2008, volume 5318 of *Lecture Notes in Computer Science*, pages 114–129. Springer Berlin Heidelberg.

Bezerra, C., Santana, F., and Freitas, F. (2014). Cqchecker: A tool to check ontologies in owl-dl using competency questions written in controlled natural language. *Learning & Nonlinear Models*, 12(2):115–129.

Blomqvist, E., Seil Sepour, A., and Presutti, V. (2012). Ontology testing - methodology and tool. In ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., and Hernandez, N., editors, *Knowledge Engineering and Knowledge Management: 18th International Conference, EKAW 2012, Galway City, Ireland*, pages 216–226, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fernández-López, M., Gómez-Pérez, A., and Juristo, N. (1997). Methontology: From ontological art towards ontological engineering. In *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series*. American Asociation for Artificial Intelligence. Ontology Engineering Group ? OEG.

García-Ramos, S., Otero, A., and Fernández-López, M. (2009). Ontologytest: A tool to evaluate ontologies through tests defined by the user. In Omatu, S., Rocha, M. P., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., and Corchado, J. M., editors, *10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops, Salamanca, Spain*, pages 91–98, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gómez-Pérez, A. (2004). *Ontology Evaluation*, chapter 13, pages 251–274. Springer Berlin Heidelberg.

Hoehndorf, R., Loebe, F., Poli, R., Herre, H., and Kelso, J. (2008). Gfo-bio: A biological core ontology. *Applied Ontology*, 3(4):219–227.

Noy, N. F. and Hafner, C. D. (1997). The state of the art in ontology design: A survey and comparative review. *AI Magazine*, 18:53–74.

Pressman, R. S. (2001). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education, 5th edition.

Schulz, S. and Boeker, M. (2013). Biotoplite: An upper level ontology for the life sciencesevolution, design and application. In *Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik angepasst an Mensch, Organisation und Umwelt, 16.-20. September 2013, Koblenz*, pages 1889–1899.

Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2012). The neon methodology for ontology engineering. In *Ontology Engineering in a Networked World.*, pages 9–34.

Uschold, M. and Gruninger, M. (1996). Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 11:93–136.

Vrandečić, D. and Gangemi, A. (2006). Unit tests for ontologies. In Meersman, R., Tari, Z., and Herrero, P., editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops: OTM Confederated International Workshops and Posters, Montpellier, France*, pages 1012–1020, Berlin, Heidelberg. Springer Berlin Heidelberg.