Incorporating Presuppositions of Competency Questions into Test-Driven Development of Ontologies

Jedrzej Potoniec[1,2], Dawid Wisniewski[1] Agnieszka Ławrynowicz[1,2]

[1] Faculty of Computing and Telecommunications, Poznan University of Technology, Poland
[2] CAMIL Center for Artificial Intelligence and Machine Learning, Poznan University of Technology, Poland

E-mail: {jpotoniec,dwisniewski,alawrynowicz}@cs.put.poznan.pl

Abstract

Ontology authoring is a complicated and error-prone process since the knowledge being modelled is expressed using logic-based formalisms, in which logical consequences of the knowledge have to be foreseen. Many approaches intended to make this task easier, use competency questions (CQs), being questions expressed in natural language to trace both the correctness and completeness of the ontology at a given time. However, CQs hold so-called presuppositions that have to be satisfied by the ontology to obtain meaningful answers from CQs. Moreover, CQs have to be expressed using a formal language, like ontology query language (SPARQL-OWL), to query the ontology. In this paper, we propose an extension of test-driven ontology development approach by formalization of presupposition satisfaction tests in terms of SPARQL-OWL queries, as well as providing translations of CQs into SPARQL-OWL queries if presupposition tests are passed. We provide a detailed description of the proposed framework and how to incorporate such tests in the workflow of test-driven development of ontologies. It is the first framework available for formalization of SPARQL-OWL queries out of CQs with their presupposition tests.

1. Introduction

Competency Questions (CQs) are questions expressed in natural language, which aim to define the scope of an ontology as part of the ontology requirements specification [14]. They are, subsequently, formalized using a relevant language, such as SPARQL [6], to automatically validate whether the ontology meets the requirements [7, 4, 2].

Ontologies can be expressed in vastly varying modeling styles, using varying modeling patterns [9], and varying features of an ontology modeling language such as the Web Ontology Language (OWL) [11]. Consider the following CQ: What software has an open source licence?. If one models different applications as different individuals, one can easily construct a suitable query in SPARQL [6] to list all instances of the class Software. However, if one models different applications as different classes, one must query for subclasses of the class Software instead.

The particular formalizations to express the tests have also varied, depending on factors such as features of the modeled domain, and included axioms and axiom sets [2], SPARQL-OWL [7, 16] or even methods based on instance assertions mimicking mock-objects [7].

In this paper, we consider formalising CQs using SPARQL-OWL [8], a variant of SPARQL with an OWL 2 DL entailment regime, that enables to handle the formal semantics of ontologies expressed in OWL 2 and assumptions underlying reasoning in OWL. Specifically, OWL makes an assumption, different from the *Closed World Assumption* made in databases: the *Open World Assumption* states that if a fact cannot be proved, it does not necessarily mean that it is false. Firstly, SPARQL-OWL will allow us to use features of the OWL and OWL reasoning. Secondly, it will also allow us to express queries with answers different than true/false, e.g., SELECT queries expressing list questions.

CQs not only encode explicit intents of their creators, but also implicit assumptions, so-called *presuppositions* [13, 4]. The notion of presuppositions comes from linguistic pragmatics, where a presupposition of a statement is a proposition whose truth is a precondition to assess whether the sentence is true or false. If the presupposition does not hold, then the sentence cannot be assessed either as true or false. Considering a question sentence, a presupposition has to be true in order for the question to have an answer. For example, the question What software has an open source licence? presupposes that software can have a licence, and if not, then the question cannot be answered.

In this paper, we provide a methodology for formalizing CQs into their corresponding formalized queries and associated presupposition queries as testing artefacts with the interpretation of their results. We consider the following sce-

DOI reference number: 10.18293/SEKE2021-165

nario: 1) Domain expert states CQs 2) Ontology engineer(s) create(s) the ontology 3) During ontology development, CQs are translated into SPARQL-OWL queries, so queries and answers can be obtained and verified after vocabulary is modelled. The scenario is analogous to software engineering, where one states unit tests before the software exists, and then, during development, these tests measure the quality of the software and may help to decide when the authoring process is complete.

We aim to address the following research questions: **RQ1**: What does it mean that a CQ is answerable? **RQ2**: What does it mean for CQ-driven ontology authoring that a presupposition is satisfied or not when it comes to testing? **RQ3**: How to handle presuppositions in the workflow of test-driven development of ontologies?

Our contributions are as follows: (a) a formalization of presuppositions using SPARQL-OWL ASK queries, (b) a model for testing list questions that considers presupposition tests with their interpretation, (c) incorporating presupposition tests into the workflow of test-driven ontology engineering, (d) a dataset of SPARQL-OWL queries enhanced with their presupposition queries.

The remainder of the paper is structured as follows. Sect. 2 describes related work. Sect. 3 introduces the formalization of presuppositions as SPARQL-OWL ASK queries, a model of testing, and describes the incorporation of presuppositions into the test-driven ontology engineering workflow. We conclude in Sect. 4.

2. Related Work

2.1. Analysis of CQs

Ren et al. [13] analysed CQs and determined patterns in the form of CQ archetypes (e.g., "Which [CE1] [OPE] [CE2]?") containing placeholders for presupposed ontology elements. Bezerra et al. [1] also proposed CQ patterns with placeholders for ontology elements, e.g. "Does <class>+ <property><class>?", functioning as Controlled Natural Language.

Wiśniewski et al. [16] and Potoniec et al. [12] analysed the natural language text of CQs itself, and a subsequent step of semantic analysis in order to find patterns. Next, they analysed the relation between the found CQ patterns and their respective SPARQL-OWL signatures (abstract representations of SPARQL-OWL query meaning), which revealed that one CQ pattern may be realized by more signatures and vice versa. Wiśniewski et al. [17] proposed a machine learning based approach to parse CQs with a model trained on over 46,000 automatically generated CQs .

Fernández-Izquierdo et al. [5] collected a corpus of ontological requirements annotated with lexico-syntactic patterns (named CORAL). The lexico-syntactic patterns have OWL constructs associated to them. These constructs were extracted from the ontology design patterns (ODPs) associated with the given lexico-syntactic pattern.

2.2. Test-driven development of ontologies

There are several tools proposed for *test-driven development (TDD)* for ontologies. Tawny-OWL [15], an ontology development framework, provides predicate functions to query the reasoner, and its answer is true/false. TDDOnto is a Protégé plugin which avails of the Protégé's syntax and uses the reasoner through the OWL API [7]. TDDOnto2, which extends TDDOnto, rigorously proves the correctness of the testing algorithms of TDDOnto, Tawny-OWL, or SCONE [3]. It generalises the algorithms of Keet and Ławrynowicz [7] to cover any OWL 2 class expression in the axiom under test.

When it comes to testing results, all the mentioned tools except TDDOnto2 give only limited information about the result of any test, being pass/fail in Tawny-OWL and SCONE. More precisely, only "axiom entailed" by the ontology is a pass and all the others statues are test failures. TDDOnto also reports missing vocabulary. TDDOnto2 specifies failure statuses more precisely, being either "inconsistent", "incoherent", or "absent".

All TDDOnto2 tests are expressed using axioms which can be tested in terms of their truth values availing of a reasoner, with a purpose of checking whether the knowledge encoded directly via the axiom is already covered in the ontology (is entailed). However, when it comes to CQs, which are questions associated often with 'gold standard' answers, they are more naturally expressed as queries, such as list queries (i.e., queries with a result being a list of objects). Therefore, in this paper, we explore such direction.

Another aspect of the mentioned tools is that they do not consider presuppositions. Indeed, binary questions (such as on the truth value of an axiom) do not have presuppositions [13] since they simply ask whether there is an answer satisfying the constraint. In this paper, contrary to the mentioned works, we consider presuppositions as we deal with SPARQL-OWL SELECT queries.

3. Incorporating presuppositions into TDD for ontologies

3.1. Presuppositions in CQs

Linguistic research on pragmatics reveals that a list question, starting with WH-words like what, which etc. always makes a presupposition that some object(s) fulfil the predicate of the question [10]. A presupposition can be generated by replacing the WH-word with the corresponding indefinite pronoun, e.g., Who left the door open? presupposes Someone left the door open [10]. It is possible to deny the presupposition, e.g., No-one left the door open [10].

From this we infer that a CQ always assumes, either explicitly or implicitly, some domain for its answer. Moreover some elements of the domain must be capable of fulfilling the predicate (*a positive presupposition*), yet the elements of the domain not necessarily fulfil the predicate (*a negative presupposition*).

3.2. Model of testing with presuppositions

Let us denote by O an OWL 2 ontology [11], and by C, D a named class or a class expression. Denote by Q a formalization of a CQ in the form of a SPARQL-OWL query, and a positive presupposition query by PQ^+, and a negative presupposition query by PQ^-. Below, we formalize *presupposition tests* availing of SPARQL-OWL queries plus the interpretation of their results.

A *presupposition query* PQ is a SPARQL-OWL ASK query with only the following basic graph pattern (BGP) in the WHERE clause: `C rdfs:subClassOf owl:Nothing`. Note, that we use such formulation since there is no direct syntax for satisfiability checking.

Definition 3.1 (Presupposition test)**.** Let $\Psi(PQ)$ denote the solution sequence, as defined by [6], of the presupposition query PQ under the OWL 2 DL entailment regime over the ontology O. If $O \models$ `C rdfs:subClassOf owl:Nothing`, then $\Psi(PQ) \neq \emptyset$ and the answer to PQ is **true**, denoted $\mu(PQ) =$ **true**, meaning the presupposition is *not satisfied*. Otherwise, $\Psi(PQ) = \emptyset$ and the answer to PQ is **false**, denoted $\mu(PQ) =$ **false**, meaning the presupposition is *satisfied*.

Furthermore, we define the model of testing for SPARQL-OWL SELECT queries Q as for those queries that have presuppositions. SPARQL-OWL ASK queries Q as corresponding to binary questions do not have presuppositions [13] since they simply ask whether there is an answer satisfying the constraint.

Let us now introduce the model of testing. We start from a CQ, for instance Which pizzas contain chocolate?. This induces a positive (informal) presupposition There may exist pizzas with chocolate, and a negative presupposition There may exist pizzas without chocolate. Both the CQ, and associated presuppositions are formalized as SPARQL-OWL queries.

Any list question (formalized as SPARQL-OWL query) can be considered as restricting a certain class expression C with another class expression D, i.e., a question about C and D. Then positive presupposition means that there are some objects that are both C and D, and if negative presupposition is satisfied, it means that there are objects that are C but not D. Non-fulfillment of a positive presupposition means that the ontology determines the answer: the intersection of C and D is necessarily an empty set; failure to meet negative presupposition means that the ontology determines the answer: the intersection of C and D is equivalent to C. If answers to all presuppositions of Q are **false**, then we can ask the query Q and interpret the obtained result.

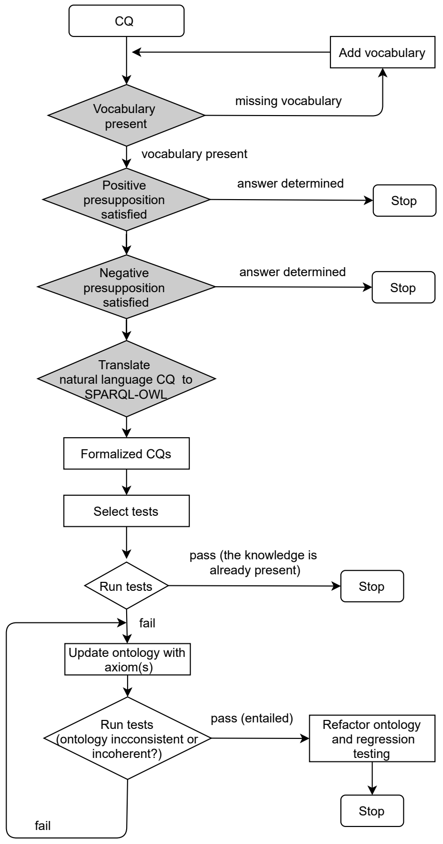
CQ
Add vocabulary
missing vocabulary
Vocabulary present
vocabulary present
Positive presupposition satisfied
answer determined
Stop
Negative presupposition satisfied
answer determined
Stop
Translate natural language CQ to SPARQL-OWL
Formalized CQs
Select tests
pass (the knowledge is already present)
Run tests
Stop
fail
Update ontology with axiom(s)
Run tests (ontology incconsistent or incoherent?)
pass (entailed)
Refactor ontology and regression testing
Stop
fail

Figure 1. Incorporating presupposition tests into the workflow of TDD for ontologies. Grey boxes represent the steps associated with our contribution, while white boxes represent the steps in the (simplified) preexisting methodology

Definition 3.2 (Model for testing (SELECT query))**.** Given a consistent and coherent ontology O, a SPARQL-OWL query Q asking about the intersection of the class expressions C_i and D_i, and its positive presuppositions PQ_i^+ (there are objects being both C_i and D_i), and a corresponding negative presuppositions PQ_i^- (there are objects being C_i, but not D_i), (i=1...n), then the result of testing Q, PQ_i^+, and PQ_i^- against O is:

$$test_O(Q) = \begin{cases} \mu(Q) = C \ \ if \ \ \exists i \ \mu(PQ_i^+) = \mathbf{true} \text{ (i,e, unsat.)} \\ \mu(Q) = \emptyset \ \ if \ \ \exists i \ \mu(PQ_i^-) = \mathbf{true} \text{ (i.e., unsat.)} \\ \text{compute the answer to } Q \ \ if \ \ \forall i \ \mu(PQ_i) = \mathbf{false} \end{cases}$$

3.3. Presuppositions in TDD workflow

Test-driven approach to ontology authoring has been shown to be theoretically and technologically a worthy solution and the recommended TDD ontology authoring workflow has been proposed [2]. Here, we extend the workflow with new additions to incorporate checking question answerability, presuppositions and list questions. The extended workflow is depicted in Fig. 1. Grey boxes represent the steps of the extended TDD workflow, which we incorporate and focus on in this paper. In particular, to determine query answerability, we not only need to check whether there is an answer to a query consistent with our intention, but whether the query can be constructed at all, including checking whether there is relevant vocabulary. Then, positive and negative presupposition tests serve to further check whether constructing a list query is meaningful.

Only after these steps, one can construct a SPARQL-OWL query out of a natural language CQ.

3.4. Dataset

In `https://tinyurl.com/3v4rfp6f` we provide a dataset constisting of SPARQL-OWL CQ query templates and their corresponding SPARQL-OWL query templates for presuppositions. The dataset is an extension of a preexisting dataset of SPARQL-OWL formalization of CQs [12, 16]. In the templates any IRIs referring to any concrete ontology are replaced by placeholders denoted by angle brackets and the same placeholder in the CQ query template and in the presuppositions templates should be replaced by the same value during materialization.

4. Discussion & conclusions

Answering RQ1, the notion of answerability of a CQ has two levels. It may seem that it is sufficient for the necessary vocabulary to be present in the ontology. While this always yields some query, and thus some answers, it fails to consider the reason for the answers, which may go against the intent of the CQ. One must thus consider the presuppositions inherent to the CQ and test them beforehand. Only when the presuppositions are satisfied the answer to the query follows the intent of the CQ.We thus claim that a CQ is (meaningfully) answerable if the necessary vocabulary is present and all the presuppositions are satisfied.

Addressing RQ2, we introduced in Sect. 3.2 the notion of a presupposition query. We formalized the way of handling such a query in order to create a presupposition test and offered guidelines to extract presuppositions from a CQ formalized as a SPARQL-OWL query.

Regarding RQ3, we extended the workflow of TDD for ontologies to incorporate presuppositions. In Sect. 3.3 we explained that an unsatisfied presupposition denotes that an answer for the CQ is predefined in a way that is incompatible with the intent of the query.

Being able to automatically provide correct SPARQL-OWL query recommendations and their presupposition tests as formalizations of ontology competency questions for a given ontology is a promising idea which can lead to reduction of time required to author the ontology. Using an already existing dataset of CQs and their translations to SPARQL-OWL helped us to get the first insight into the problem. We hope that our model for extending TDD for ontologies with presupposition tests, together with a new dataset, will incent further research into more commonsense-aware knowledge engineering.

References

[1] C. Bezerra et al. CQChecker: A tool to check ontologies in OWL-DL using competency questions written in controlled natural language. *L&NLM*, vol. 12:pp. 115–129, 01 2014.

[2] K. Davies et al. More effective ontology authoring with test-driven development and the TDDonto2 tool. *IJAIT*, vol. 28(7):pp. 1950023:1–1950023:25, 2019.

[3] R. Denaux, D. Thakker, V. Dimitrova, and A. G. Cohn. Interactive semantic feedback for intuitive ontology authoring. In *Proc. of FOIS'12*, pages 160–173. IOS Press, 2012.

[4] M. Dennis et al. Computing authoring tests from competency questions: Experimental validation. In *Proc. of ISWC*, pages 243–259, 2017.

[5] A. Fernández-Izquierdo et al. CORAL: A corpus of ontological requirements annotated with lexico-syntactic patterns. In *Proc. of ESWC*, LNCS, pages 443–458. Springer, 2019.

[6] S. Harris and A. Seaborne. SPARQL 1.1 query language. W3C recommendation, W3C, Mar. 2013.

[7] C. M. Keet and A. Lawrynowicz. Test-driven development of ontologies. In *ESWC*, pages 642–657. Springer, 2016.

[8] I. Kollia et al. SPARQL query answering over OWL ontologies. In *Proc. of ESWC, Part I*, pages 382–396, 2011.

[9] A. Lawrynowicz et al. Discovery of emerging design patterns in ontologies using tree mining. *Semantic Web*, vol. 9(4):pp. 517–544, 2018.

[10] J. Lyons. *Semantics, vol. 2*. Cambridge Univ. Press, 1977.

[11] B. Motik, P. F. Patel-Schneider, and B. Parsia. OWL 2 web ontology language structural specification and functional-style syntax (second edition). Technical report, W3C, 2012.

[12] J. Potoniec et al. Dataset of ontology competency questions to sparql-owl queries translations. *Data in Brief*, 29, 2020.

[13] Y. Ren et al. Towards competency question-driven ontology authoring. In *ESWC*, pages 752–767. Springer, 2014.

[14] M. C. Suárez-Figueroa and A. Gómez-Pérez. Ontology requirements specification. In *Ontology Engineering in a Networked World*, pages 93–106. Springer, 2012.

[15] J. D. Warrender and P. Lord. How, what and why to test an ontology. *CoRR*, abs/1505.04112, 2015.

[16] D. Wisniewski et al. Analysis of ontology competency questions and their formalizations in SPARQL-OWL. *JWS*, 59, 2019.

[17] D. Wisniewski and A. Ławrynowicz. A tagger for glossary of terms extraction from ontology competency questions. In *ESWC, Satellite Events*, pages 181–185. Springer, 2019.