OntoComP: A Protégé Plugin for Completing OWL Ontologies

Barış Sertkaya*

Theoretical Computer Science, TU Dresden, Germany
sertkaya@tcs.inf.tu-dresden.de

Abstract. We describe OntoComP, a Protégé 4 plugin that supports ontology engineers in completing OWL ontologies. More precisely, OntoComP supports an ontology engineer in checking whether an ontology contains all the relevant information about the application domain, and in extending the ontology appropriately if this is not the case. It acquires complete knowledge about the application domain efficiently by asking successive questions to the ontology engineer. By using novel techniques from Formal Concept Analysis, it ensures that, on the one hand, the interaction with the ontology engineer is kept to a minimum, and, on the other hand, the resulting ontology is complete in a certain well-defined sense.

1 Introduction

Ontologies play a key role for the Semantic Web. Since the standardization of OWL as the ontology language for the Semantic Web, several ontology editors now support OWL [4,3], and ontologies written in OWL are employed in more and more semantic web applications in various domains. As the number and size of these ontologies grows, tools that support improving their quality become more important. The tools available until now mostly deal with detecting inconsistencies and inferring consequences, i.e., implicit knowledge that can be deduced from the knowledge explicitly represented in the ontology. There are also promising approaches that allow to pinpoint the reasons for inconsistencies and for certain unwanted consequences. These approaches address the quality dimension of *soundness* of an ontology, both within itself (consistency) and w.r.t. the intended application domain (no unwanted consequences). In our previous work [1,5], we have considered a different quality dimension, namely *completeness* of the knowledge in an ontology. We have provided a formally well-founded technique called *ontology completion*, that supports the ontology engineer in checking whether an ontology contains all the relevant information about the application domain, and in extending the ontology appropriately if this is not the case.

An OWL ontology typically consists of two parts, the terminological part (TBox), which defines concepts and also states additional constraints (so-called

* Supported by German Research Foundation (DFG) under grant BA 1122/12-1.

L. Aroyo et al. (Eds.): ESWC 2009, LNCS 5554, pp. 898–902, 2009.

general concept inclusions or GCIs) on the interpretation of these concepts, and the assertional part (ABox), which describes individuals and their relationship to each other and to concepts. Given an application domain and an OWL ontology describing it we can ask:

- Are all the relevant constraints that hold between concepts in the domain captured by the TBox?
- Are all the relevant individuals existing in the domain represented in the ABox?

Such questions cannot be answered by an automated tool alone. Clearly, to check whether a given relationship between concepts—which does not already follow from the TBox—holds in the domain, one needs to ask a domain expert, and the same is true for questions regarding the existence of individuals not described in the ABox. The role of the ontology completion tool here is to ensure that the expert is asked as few questions as possible; in particular, she should not be asked trivial questions, i.e., questions that could actually be answered based on the represented knowledge.

2 Motivating Example

As an example, of how ontology completion supports the ontology engineer in practice, consider the OWL ontology for human protein phosphatases that has been described and used in [7]. This ontology was developed based on information from peer-reviewed publications. The human protein phosphatase family has been well characterised experimentally, and detailed knowledge about different classes of such proteins is available. This knowledge is represented in the terminological part of the ontology. Moreover, a large set of human phosphatases has been identified and documented by expert biologists. These are described as individuals in the assertional part of the ontology. One can now ask whether the information about protein phosphatases contained in this ontology is complete. That is, are all the relationships that hold among the introduced classes of phosphatases captured by the constraints in the TBox, or are there relationships that hold in the domain, but do not follow from the TBox? Are all possible kinds of human protein phosphatases represented by individuals in the ABox, or are there phosphatases that have not yet been included in the ontology or even not yet been identified?

Clearly, these questions need to be answered by a biologist. In this example, answering a non-trivial question regarding human protein phosphatases may require the biologist to study the relevant literature, query existing protein databases, or even to carry out new experiments. Thus, the expert may be prompted to acquire new biological knowledge.

3 Ontology Completion

The key technologies lying under ontology completion are Description Logic reasoning, and the attribute exploration method developed in Formal Concept

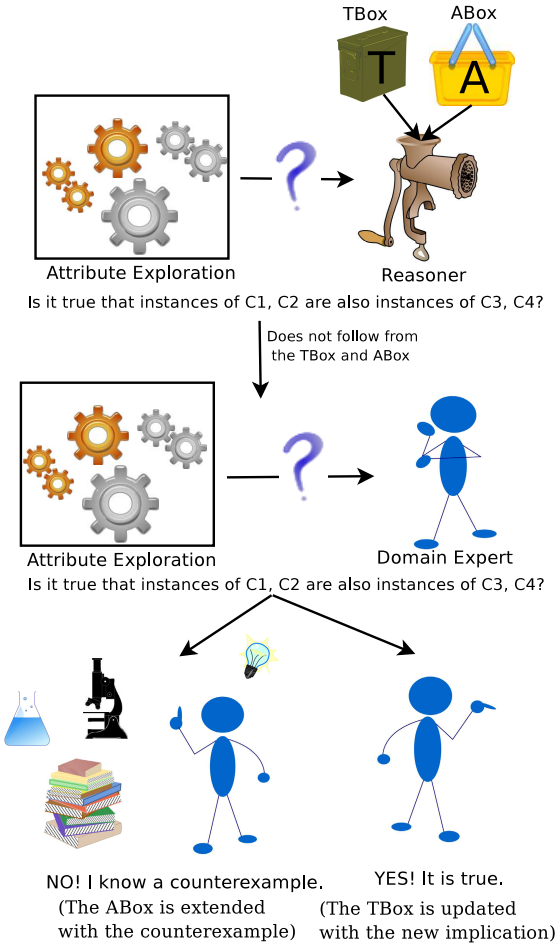

Fig. 1. Ontology completion process

Analysis (FCA) [2]. FCA is a field of applied mathematics that aims to formalize the notions of a concept and a conceptual hierarchy by means of mathematical tools. It is used for conceptual data analysis and knowledge processing. Attribute exploration is a knowledge acquisition method of FCA that is used to acquire complete knowledge about an application domain by asking successive questions to a domain expert. It asks the expert questions of the form "*is it true that instances of the classes* $C_{i1}, \ldots, C_{ik}$ *also instances of* $C_{j1}, \ldots, C_{jl}$*?*". When such a question is asked, the expert is expected to either confirm the question, in which case a new implication in the application domain has been discovered, or reject it. If the expert rejects such a question, she is expected to give a counterexample, i.e., an individual that is instances of the classes $C_{i1}, \ldots, C_{ik}$ but is not an instance of at least one of $C_{j1}, \ldots, C_{jl}$. This counterexample is then added to the ontology as a new individual, and the next question is asked.

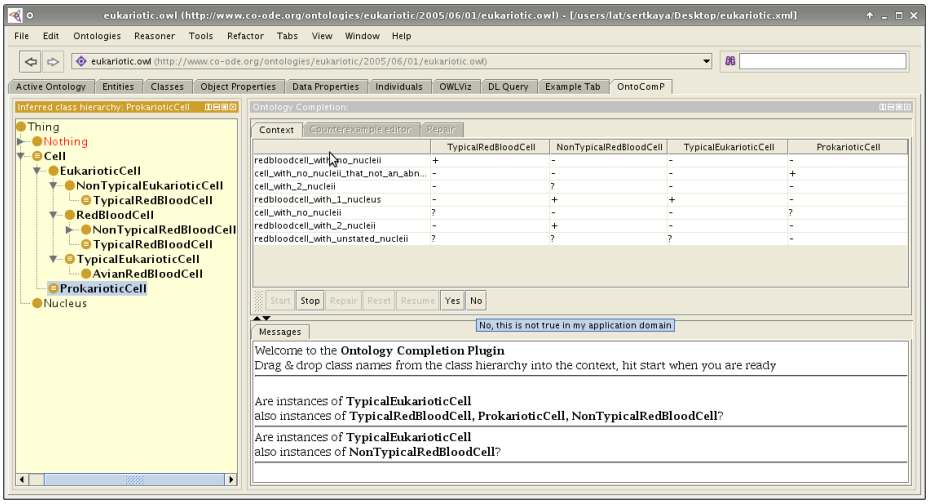

Fig. 2. OntoComP window during completion

What makes attribute exploration an attractive method for capturing expert knowledge is that it guarantees to make best use of the expert's answers, and to ask the minimum possible number of questions that suffices to acquire complete knowledge about the application domain.

4 OntoComP

Based on our results in [1,5], we have implemented an open-source ontology completion tool called OntoComP[1], which stands for ONTOlogy COMpletion Plugin. It is written in the Java programming language as a plugin for the Protégé 4 ontology editor. It can be easily installed by just copying the `jar` file provided under the URL given below into the `plugins` directory of an existing Protégé 4 installation. Then upon a new start, Protégé will find the OntoComP plugin and open a new tab for it.

In order to complete an ontology with OntoComP, the user first classifies the ontology with a reasoner that is supported by Protégé 4, namely FaCT++ or Pellet. Once the ontology is classified, the OntoComP tab displays the class hierarchy on the left. At this point the user can drag "interesting" class names, which she wants to have in the completion process, from this hierarchy and drop them into the `Context` tab on the right. Simultaneously, the instances of these classes will also be displayed in a table in the `Context` tab. In this table a `+` in row a and column C means that the individual a is an intance of the class C, a `-` means that a is an instance of the complement of C, and a `?` means that nothing is known about the membership of a in class C. When the user is done with selecting the relevant classes, she starts the completion by hitting

[1] Available under `http://ontocomp.googlecode.com`

the *Start* button and sees the first question in the `Messages` tab. If she confirms the question by hitting the *Yes* button, OntoComP comes up with the next question. If she rejects the question by hitting *No* button OntoComP opens the `Counterexample editor` tab where the user can generate a counterexample to the rejected question. The process continues until all questions have been answered by the user. During completion, at any time the user can suspend the completion process and see her answering history. If she thinks she has made an error at some point, she can repair it by undoing that particular erroneous answer and can continue completion. A similar FCA-based ontology refinement tool that is related to OntoComP, but mainly aimed at acquiring domain-range restrictions has been described in [6].

5 The Demo

During the demo, the audience will have the opportunity to see OntoComP in action on ontologies from real world application domains, which do not require specialized expert knowledge (so that the demo is still understandable for the whole audience). We are going to demonstrate in detail how to use OntoComP and how it supports the ontology engineer in completing an ontology. We are going to prepare scenarios where the ontology engineer makes errors during ontology completion and demonstrate how to repair them.

References

1. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing Description Logic Knowledge Bases using Formal Concept Analysis. In: Proc. of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007). AAAI Press, Menlo Park (2007)
2. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
3. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., Hendler, J.A.: Swoop: A web ontology editing browser. Journal of Web Semantics 4(2), 144–153 (2006)
4. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The protégé OWL plugin: An open development environment for semantic web applications. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 229–243. Springer, Heidelberg (2004)
5. Sertkaya, B.: Formal Concept Analysis Methods for Description Logics. Ph.D. dissertation, Institute of Theoretical Computer Science, TU Dresden, Germany (2007)
6. Völker, J., Rudolph, S.: Fostering web intelligence by semi-automatic OWL ontology refinement. In: Proc. of the 7th International Conference on Web Intelligence (WI 2008) (2008)
7. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P.W., Sattler, U., Turi, D., Stevens, R.: A little semantic web goes a long way in biology. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 786–800. Springer, Heidelberg (2005)