Assessing the Safety of Knowledge Patterns in OWL Ontologies

Luigi Iannone, Ignazio Palmisano, Alan L. Rector, and Robert Stevens

University of Manchester
Kilburn Building
Oxford Road M13 9PL
Manchester, UK
lastname@cs.manchester.ac.uk

Abstract. The availability of a concrete language for embedding knowledge patterns inside OWL ontologies makes it possible to analyze their impact on the semantics when applied to the ontologies themselves. Starting from recent results available in the literature, this work proposes a sufficient condition for identifying *safe* patterns encoded in OPPL. The resulting framework can be used to implement OWL ontology engineering tools that help knowledge engineers to understand the level of extensibility of their models as well as pattern users to determine what are the safe ways of utilizing a pattern in their ontologies.

1 Introduction

OWL ontologies may be built for the most disparate purposes. Yet, as soon as they are made public, the problem of their extension arises. The presence of the `owl:import` primitive in the language testifies to the inclination of OWL to encourage ontology re-use. The addition of any axiom to an ontology does, however, modify its semantics. The extent of the impact of these alterations can be evaluated, but , at the moment, not anticipated or formally analyzed when an ontology is either being developed or recently released. The knowledge engineers, at the current state of the art, have only annotations for documenting what are the possible extensions of their ontology, and there is no formal language for their encoding.

The effect of this limitation is better explained using an analogy. Let us suppose for a moment that the Java programming language did not provide the `final` primitive to specify that a certain Java class/method cannot be sub-classed/overridden. API developers could not prevent their users from extending portions of the API object model that are meant to be fixed and non extensible. The consequence would be that all the assumptions about some pieces of code behaving in a fixed and pre-determined way would no longer hold, thus making the re-use of third party code more unpredictable and unreliable. One could argue that knowledge re-use is meant to be more flexible than the code counterpart, that is why we observe that nothing as strict as the `final` primitive is needed. What we claim, though, is that it would be useful if the knowledge engineer, whilst developing their ontologies, could specify what and how to extend it in order to remain compliant with its original meta-model. The users could then decide whether to follow such guidelines or not, but natural language annotations are too

L. Aroyo et al. (Eds.): ESWC 2010, Part I, LNCS 6088, pp. 137–151, 2010.