Development of a server to support the formal semantic web query language OWL-QL

Jan Galinski, Atila Kaya, Ralf Möller
Hamburg University of Science and Technology
Hamburg, DE
mail@jan-galinski.de, at.kaya|r.f.moeller@tuhh.de

In this abstract we present a server that supports OWL-QL query-answering dialogues by making use of the reasoning capabilities of the DL reasoner Racer. Currently, users can use Racer to query an OWL knowledge base using nRQL or DIG. Our primary goal in developing an OWL-QL server is to offer as many as possible reasoning functions of Racer in the standard query language OWL-QL.

This first version of our implementation is aimed to support significant OWL-QL features that can easily be translated into the native Racer Query Language nRQL. Currently, it supports queries for ABox retrieval with distinguished variables (must-binds) and conjunctive queries. Further important OWL-QL features, such as may-bind variables, can be expressed in nRQL through reformulation and will be supported in the future. The response collections returned by the server contain no duplicate answers. Moreover the server provides non-redundant answers.

In order to develop such a server we enhanced the existing Racer proxy to offer OWL-QL support as a web service. The existing Racer proxy is a standalone java application that serves as a Racer front-end for clients. They can communicate with the Racer proxy in nRQL or DIG. We integrated standard open source software components and frameworks into the architecture, such as Tomcat, Apache Axis, the XMLBeans Framework and the Jena Semantic Web Framework, so that the solution gets more transparent and will benefit from further component evolution.

Our solution offers OWL-QL support as a web service in order to enable the interaction of clients and server, which are loosely coupled. A pivotal feature of our solution is a caching mechanism that operates at two different layers of the architecture: At the bottom layer of the architecture the iterative query answering support of the new Racer version (tuple-at-a-time mode) is used to cache results. Secondly, at the top layer we implemented a caching mechanism in OWL-QL dialogues that caches queries, answers and client session information.