Brain, a library for the OWL2 EL profile

Samuel Croset[1], John Overington[1], and Dietrich Rebholz-Schuhmann[1]

EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD UK
croset@ebi.ac.uk

Abstract. Brain is a Java library facilitating the interaction with OWL2 EL ontologies. The library aims at bridging the gap between graphical user interfaces (GUI) such as Protege and the OWL-API: It provides a series of convenience methods to create and query knowledge bases using the Manchester syntax. The library is useful to develop web applications and particularly suited for the biomedical domain. Brain relies on ELK for reasoning tasks. The open source project is available at https://github.com/loopasam/Brain.

Keywords: OWL2 EL, library, Java, Manchester syntax

1 Introduction

1.1 OWL2 EL

The second version of the Web Ontology Language (OWL2) introduces a series of profiles: OWL2 EL, RL and QL. These profiles are subsets of the full OWL2 specification and have been designed to match the requirements of particular case scenarios. Profiles are defined by the type of axioms and constructs they support: For instance the OWL2 EL profile, which has been created for the biomedical domain, does not allow disjunctions or cardinality restrictions. This limited expressivity however enables the implementation of fast reasoning algorithms which can handle large data as it is required by the application domain. In this document we will present the core features of a Java library, Brain, dedicated to support the OWL2 EL families.

1.2 Motivation

The biomedical domain is particularly interesting for the OWL community because of the richness and variety of the ontologies it contains. The majority of them such as the Gene Ontology (GO) or SNOMED CT are very large but they are fortunately following the EL profile, which enables the use of recent and fast reasoners such as ELK. Because of these improvements in the reasoning speed over big biomedical knowledge bases, it becomes nowadays possible to build web applications or to perform biological analysis relying heavely of OWL queries performed against an ontology of interest. These taks are traditionnaly done either via a graphical user interface (GUI) such as Protege or programmed using the OWL-API and the Java language. GUIs are wonderful to develop toy examples, but they are not suited to develop very large ontologies as the ones faced in life sciences, potentially containing millions of axioms. The OWL-API is solid solution to build applications but it could be daunting for new comers or users with little experience in Java. OWLTools is an intermediary solution for the biomedical domain, but the interaction with this library is mostly done by command lines with impairs the developement of larger projects. In order to address these issues, we have developped Brain, a facade on the top of the OWL-API. The library facilitates the manipulation of OWL2 EL ontologies, specially in a web server setting. Brain is already used in production as a back-end engine for web applications such as the Virtual Fly Brain or the Functional Therapeutic Chemical Classification System.

2 Features

2.1 Availability

maven, jar with dependencies code source on github

presentation of the core features of the lib

Unique ontology

Basically, a instance of a Brain object hold a reference to only one ontology (also called knowledge-base). You can of course import some external ontologies, refer to external terms, but at the end it will resolve to only one ontology.

Unique names

The names of OWL entities handled by a Brain object have to be unique. This is motivated by the fact that Brain hides as much as possible the cumbersome interaction with prefixes, IRI and URIs. Everything resolves to names.

Typeless

The interaction with the ontology is done via strings (type-less). Expressions and queries are formulated in Manchester syntax. It results in less and more explicit code.

Error-handling driven

Because the interaction with Brain is built around strings rather than Java objects, special care has to be put on exception handling, in order to preserve the consistency. Brain throws a lot of possible exceptions, depending the type of operation you want to carry. All exceptions are subclasses of BrainException.

Queries

integration

3 Example of implementation

nucleus axiom longer

VBF use case + FTC

4 Conclusion

future direction –¿ SVG graphs in dev, add individuals, etc..

References

1. Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. 78, 315–333 (1982)
2. Clarke, F., Ekeland, I.: Solutions périodiques, du période donnée, des équations hamiltoniennes. Note CRAS Paris 287, 1013–1015 (1978)
3. Michalek, R., Tarantello, G.: Subharmonic solutions with prescribed minimal period for nonautonomous Hamiltonian systems. J. Diff. Eq. 72, 28–55 (1988)
4. Tarantello, G.: Subharmonic solutions for Hamiltonian systems via a $\mathbb{Z}_p$ pseudoindex theory. Annali di Matematica Pura (to appear)
5. Rabinowitz, P.: On subharmonic solutions of a Hamiltonian system. Comm. Pure Appl. Math. 33, 609–633 (1980)