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 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 well 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 and oriented toward biomedical knowledge manipulation.

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 resources such as the Gene Ontology (GO) or SNOMED CT are very large but they are fortunately in line with the EL profile. Driven by the recent improvements in the reasoning speed over big biomedical knowledge bases, with reasoners such as ELK, it becomes nowadays possible to build web applications or to perform biological analysis relying heavily on OWL queries performed against an ontology of interest. The options available to undertake such tasks span from graphical user interfaces (GUI) such as Protege to application programming interfaces as the OWL-API. GUIs are wonderful to develop toy examples, but they

are not suited to handle 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 ease the interaction with OWL2 EL we have developped Brain, a facade on the top of the OWL-API. The library facilitates the manipulation of EL ontologies, specially in a web server setting. Brain is already successfully 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

The rest of the document will present the core features of Brain as well as an example of axiom implementation. The main idea behind Brain is to prevent ambiguous situations to happen while creating an ontology and to simplify the interaction with biomedical knowledge bases. Brain is implemented following a pattern facade: It wraps the interaction with the OWL-API and the ELK reasoner by providing some convenience methods. The wrapped softwares yet remain accessible for operation un-implemented by Brain.

2.1 Availability

The source code of the library is open and freely available at https://github.com/loopasam/Brain/ under an Apache License 2.0. Brain binaries are also distributed on Maven or directly dowloadable as a jar file including all the dependencies. The documentation providing concrete examples can be accessed and edited at https://github.com/loopasam/Brain/wiki.

2.2 Manchester syntax interaction

The interaction with Brain resolves only to the Manchester syntax expressed as Java strings. This notation is user friendly and already widely used inside Protege. It allows to focus on the OWL axioms rather than on the programming. The advantage of working with strings as input rather than with more complicated Java objects appears in a web application context: HTTP messaging can indeed be easily parsed and interpreted in order to quickly build an OWL endpoint for instance. A list of OWL queries for instance is also easier to maintain, read and understand when represented with the Manchester syntax.

2.3 Unique ontology

One of the core design decision behind the library was to enforce the users to work with only one ontology. This simplification helps the user to focus on one knowledge base at the time and remove any ambiguity that could be present

while dealing with scattered entities. It is yet possible to import and load external ontologies and entities, but Brain merge any new axioms into one single knowledge base.

2.4 Unique short form names

Brain also enforces OWL entities to have a unique short form names. This choice was made in order to prevent ambiguity while using the Manchester syntax to formulate OWL expressions. For instance the short forms of the OWL entities http://example.org/Cat and http://semanticweb.org/Cat will be the same: Cat. It leads to problems when an expression is formulated with the Manchester syntax later on using the short form name. The entity Cat is indeed ambigous in this case. This issue is avoided by Brain and an exception is thrown when a potentially dangerous situation occurs. This choice is not exactly inline with some of the Semantic Web philosophy, but it seems likely to work in the biomedical domain, where entity identifiers are most of the time preceded by a unique prefix. For example the short form names of the Gene Ontology classes all start with the prefix "GO_", uniquely identifying the biomedical resource. Note that this design is not against the unique name assumption: It is still possible that different names refer to the same entities.

2.5 Error-handling driven

As presented before, Brain enforces some constraints in order to prevent some errors with OWL logic later on. A lot of different types of errors can be thrown in order to alert the user that a problem exists in the underlying ontology. A special care had also to be put when interacting with expressions entered as strings. Problems encoutered when creating or querying the ontology are made explicit to the developer in order to preserve the consistency of the underlying knowledge base.

2.6 Queries

A common use-case for OWL ontologies is running queries over them in order to retrieve some implicit information. An OWL query is a standard OWL class expression in the context of Brain. The ELK reasoner is handling classification and query tasks. The classification of the ontology is triggered only when absolutely needed and transparent to the user (lazy initialization). Queries are also formulated using the Manchester syntax and it is possible to use the label (rdfs:label) of OWL entities to enter the query expression, as featured in Protege. Queries are thread-safe: For instance the simultaneous queries of two users will be handled by Brain. This characteristic is important in concurrent web server setting for instance.

2.7 Limitations

The methods present in the library are strongly driven from real world use-cases and requirements from users. However some important features are still missing: Individuals are not currently supported for example. Despite not being oriented towards OWL individuals, OWL2 EL ontologies are yet sometimes using them. This drawback will be addressed in a coming release.

3 Example of implementation

The easiest way to understand the interaction with Brain is probably by looking at a concrete example. This section will present the implementation of a biomedical axiom using the library. The features discussed before are illustrated in Figure 1. The interested reader can find more examples at https://github.com/loopasam/Brain/wiki/Examples.

4 Conclusion

Brain is a Java library leveraging the interation with the OWL-API and the ELK reasoner. The library is dedicated to OWL2 EL ontologies and provide an interface to manipulate and query biomedical knowledge bases. The library has been designed to build web applications and to perform biomedical analysis relying heavely on a reasoner. Some features will be added in future releases, such as individual support. New functionnalities are also currently implemented, such as the generation of scalable vector graphics (SVG) representing the OWL class taxonomies from the underlying ontology. Brain is already stably used in some real life biomedical applications and new use-cases from the biomedical community will guide the further developements of the library.

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)

```
//Creation of a fresh Brain instance.
//All the operations will be done via this object.
Brain brain = new Brain();

//Add some OWL classes to the ontology.
//Short forms or full URIs can be used, but
//the short forms have to be unique.
brain.addClass("Nucleus");
brain.addClass("http://example.org/Cell");

//Add an OWL object property to the ontology
brain.addObjectProperty("part-of");

//Declare a complex axiom (partial existential restriction).
//Note that OWL expressions in
//Manchester syntax can be used directly for the axiom
//assertion. If the classes used in the expression were
//missing, an error will be thrown (not shown here).
brain.subClassOf("Nucleus", "part-of some Cell");

//Integrate the content of an external knowledge base
//with the current ontology.
brain.learn("http://example.org/bar.owl");

//Query the knowledge base for indirect subclasses
//with the help of the ELK reasoner.
//Note that the classification initialization is
//transparent to the user.
List<String> subClasses =
                brain.getSubClasses("part-of some Cell", false);

//Free the resources cached by the reasoner.
//Due to ELK implementation, allows
//efficient multi-threaded reasoning.
brain.sleep();

//Save the ontology as a file
//using the Manchester syntax.
brain.save("your/path/to/ontology.owl");
```

Fig. 1. Implementation example in Java of an axiom using Brain; the axiom expressed in natural language: *A nucleus is part of some cells.* Same axiom described in OWL using the Manchester syntax: *Nucleus subClassOf part-of some Cell.*