BIOINFORMATICS

Vol. 00 no. 00 2005
Pages 1–2

Brain: Biomedical Knowledge Manipulation

Samuel Croset [1,*], Robert Hoehndorf [2], John Overington [1] and Dietrich Rebholz-Schuhmann [1]

[1]European Bioinformatics Institute, Wellcome Trust Genome Campus, Cambridge, CB10 1SD UK
[2]Department of Genetics, University of Cambridge, Downing Street, Cambridge, CB2 3EH, UK

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Summary: Brain is a library optimizing the creation and manipulation of OWL 2EL ontologies and knowledge bases. It relies on the OWL-API and uses Elk for reasoning tasks.

Availability and Implementation: The Java source code and the library are freely available at https://github.com/loopasam/Brain and on the Maven Central repository (GroupId: uk.ac.ebi.brain). The documentation is available at https://github.com/loopasam/Brain/wiki.

Contact: croset@ebi.ac.uk

Supplementary information: Links to additional figures/data available on a web site, or reference to online-only Supplementary data available at the journal's web site.

Relational databases hold most of the available structured biomedical information. The content of these repositories is often extracted from scientific literature by manual curation with the help of text-mining tools. The transformation from raw text into structured data is most important, as the curated information can then be classified, managed and queried more easily. Databases facilitate the re-use of previous work in a computer-friendly manner and support the biomedical knowledge to scale-up. In order to leverage further more the existing information, the current trend is at data integration and interoperability, with large projects such as ELIXIR leading the way. The underlying idea assumes that increasingly complex biomedical challenges such as finding new treatments for diseases could be addressed by combining the content of independant repositories via the Internet. Traditional relational databases are however an obstacle to realize this vision, mostly because of their lack of support for interoperability: The schema structuring the data is indeed very repository-specific which limits the combination of the native content with external data in an efficient and meaningful fashion. In order to address this issue, a series a standard forming the semantic web have been developped. One of them, the Resources Description Framework (RDF) enables the exposition of the underlying structure as part of the data themselves. This representation relies on triples as building block, composed as *subject - relation - object* and where the types of the data are identified with a special relation called *rdf:type*. More complicated data structures, such as sub-classes or transitive relations can be further expressed via another standard, the Web Ontology Language (OWL). OWL derives from Description logic and is used to capture the knowledge of a domain of interest in the form of a structured vocabulary. This feature makes it particularly interesting from the point of view of life science, as a lot of ontologies and classification have been develloped since the origin of the discipline. OWL is often expressed in combination with RDF data in order to reveal the underlying schema, but it can also be used as such as an implemetation of a part of description logic. Knowledge bases and ontologies can therefore be built without being necessarly expressed as RDF triples while still preserving all the advantages in regards to data integration and interoperability. Brain, the library presented in this manuscript aims at facilitating the contruction and manipulation of such knowledge bases (refered in the manuscript also as ontologies), rather than being oriented towards the consumption of RDF data. We will present first the biomedical motivation for the particular subset of OWL supported by Brain so called OWL 2EL. The will be discussed the main features implemented by Brain in regards to this profile.

OWL 2EL is a profile of OWL as it features only a subset of the constructs available in the original language. This profile is designed to be *tractable*, meaning that the constructs available have a polynomial complexity which is easier to compute than the full version of OWL. These constructs are called *axioms* and are the fundamental block behind OWL knowledge bases. Axioms assert the facts and relations present in the knowledge base and are understood by a computer program named *reasoner*. Based on the logical structure of the axioms, the reasoner is indeed capable of deriving new facts from the asserted ones as well as retrieving some implicit information, enabling powerful query mecanisms surpassing the Structured Query Language (SQL) expressivity. A reasoner can also check the consistency of an ontology and report back the axioms violating a particular profile. Because of these computational properties, OWL 2EL is suitable for real-life bomedical applications, where millions of axioms are potentially present. Moreover, the profile is expressive enough for a good portion of the biomedical knowledge: Most of the ontologies such as the Gene Ontology or Chebi are already included in this profile and any relational model can be easely converted and represented using OWL 2EL. OWL knowledge representation also separates *individuals* known as Assertional Box (ABox) from *classes*, the Terminological Box (TBox). In life science, most of the entries of databases describing molecular concepts are to be considered as classes rather than instances, as they describe the generic version of a molecular entity, such as a protein which has a lot of materialization in practice, namely the actual proteins. This

*to whom correspondence should be addressed representation differs a lot from the one used in the relational model or with RDF, where entries are considered as instance records rather than as actual biological objects. While working with OWL this distinction is important to make, as it influences the inferences mades by the reasoner. Fortunately OWL 2EL focuses on this aspect and enables the handling of knowledge bases with a large number of classes. Another important feature of OWL 2EL comes from the possibility to run some reasoning tasks in parallel. Clusters or multicore settings can therefore scale the speed of reasoning as more data are added to the knowledge base.

In practice, the implementation of OWL ontologies and knowledge bases can be done either in a programmatic way via the OWL-API or with the help of a visual tool such as Protege. The graphical interface of Protege allows the user to focus more on the OWL constructs themselves rather than on the programmatic implementation. OWL axioms and expressions can indeed be entered with the user-friendly Manchester syntax and the name of the classes are displayed without their URL prefixes for instance. Applications requiring a deeper control over the ontology can employ the OWL-API itself but it requires a fairly good understanding of Java. The API deals in great details with all the aspects around ontology generation but the level of detail can be cumbersome for some applications. Brain aims at filling the gap between the OWL-API and graphical interfaces: It is designed as a facade, providing a series of convenience methods for the common use-cases. Table 1 highlights such constructs, with their equivalent in description logic and OWL 2EL.

Table 1. Example of some common OWL 2EL constructs written using the Manchester syntax alonside an example of implementation using Brain.

OWL	Brain implementation
Class: A	brain.addClass("A");
ObjectProperty: P	brain.addObjectProperty("P");
C and D	brain.getEquivalentClasses("C and D");
owl:Thing	brain.getOWLClass("Thing");
P some C	brain.subClassOf("X", "P some C");
C SubClassOf: D	brain.subClassOf("C", "D");
S SubPropertyChain: R1 o R2	brain.chain("R1 o R2", "S");
R Characteristics: Transitive	brain.transitive("R");

Brain builds on the top of the fast Elk reasonner specialised for OWL 2EL ontologies for reasoning tasks. Elk shows very good performances at classifying large datasets and ontologies and is implemented in a multi-thread friendly way. Brain relies on the intuitive Manchester syntax to formulate queries and class expressions. The interaction with the OWL-API resolves around strings, making Brain suitable to parse and answer requests in the context of a web service or an OWL end-point for instance. The library supports the loading and referring of external ontologies or knowledge bases in order to integrate and reason in a scalable way over data coming from different sources, such as independant biomedical repositories. Conclusion Why is brain interesting for the community tool to build KB scalable solution in order to preapre the next stage of data integration with KB reasoner driven and semantic web.

ACKNOWLEDGEMENT

blablaba

Funding: This work was supported by

REFERENCES