BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 18 2010, pages 2357–2358
doi:10.1093/bioinformatics/btq416

Databases and ontologies

Advance Access publication on July 13, 2010

OWL2Perl: creating Perl modules from OWL class definitions

Edward Kawas* and Mark D. Wilkinson*

Heart + Lung Institute at St. Paul's Hospital, Vancouver, BC V6Z 1Y6, Canada

Associate Editor: John Quankenbush

ABSTRACT

Summary: Support for utilizing OWL ontologies in Perl is extremely limited, despite the growing importance of the Semantic Web in Healthcare and Life Sciences. Here, we present a Perl framework that generates Perl modules based on OWL Class definitions. These modules can then be used by other software applications to create resource description framework (RDF) data compliant with these OWL models.

Availability: OWL2Perl is available for download from CPAN, under the module name OWL2Perl. It is released under the new BSD license.

Contact: edward.kawas@gmail.com; markw@illuminae.com

Received on April 28, 2010; revised on June 9, 2010; accepted on July 9, 2010

1 INTRODUCTION

The Semantic Web can be envisioned as two layers—linked data and explicit knowledge—encoded in two technologies, Resource Description Framework (RDF; http://www.w3c.org/RDF) and Web Ontology Language (OWL; http://www.w3.org/TR/owl-features/) respectively. In RDF, data points or data entities are named by Universal Resource Indicators (URIs), and pairs of URIs are linked together with meaningful predicates that describe the relationship between the two URIs. The resulting network or 'graph' is then interpreted by knowledge encoded in OWL ontologies; the URI's become explicitly 'typed' as instances of ontologically defined classes based on the network of predicates and values surrounding them. Thus, a URI with the predicates 'color', 'texture', 'taste', 'weight' with values 'red', 'crunchy', 'sweet', '300 g' might be classified as type 'Apple' by a fruit ontology, or as 'basketable-item' by a gift-basket ontology.

Though the above is a scenario in which ontologies are used to interpret and classify existing data, ontologies can also be used to guide the creation of structured data, in a manner akin to how an XML schema is used to constrain the structure of a valid XML document. This is useful as it allows one to generate and publish data compliant with some defined knowledge structure in order to facilitate other's integration and interpretation of that data.

The bioinformatics community is rapidly adopting Semantic Web technologies, though there is a wide variation in the extent to which formal logics have been adopted by the various projects. At one end of the spectrum, perhaps the most recognized community is the Open Biological Ontologies project (OBO; Smith *et al.*, 2007). OBO ontologies are primarily declarative class hierarchies, where the conditions for class membership are not part of the 'computable' definition of the class. At the other end of the spectrum, are ontologies such as PhosphaBase (Wolstencroft *et al.*, 2005) and the Lipid Ontology (Baker *et al.*, 2008) where each class is axiomatically defined by the properties and property values that are required by members of that class. It is these logically rich ontologies that we aim to support with OWL2Perl.

Support for both RDF and OWL is strong in the Java programming language. In the Perl programming language, there are several projects that provide support for RDF, including RDF::Core (http://search.cpan.org/~dpokorny/RDF-Core/); however Perl support for OWL is extremely limited. ONTO-Perl (Antezana *et al.*, 2008) and go-perl (http://search.cpan.org/~cmungall/go-perl/) both handle OBO-style ontologies, but cannot consume the full breadth of OWL logical constructs. Moreover, these two projects are primarily aimed at manipulating the ontology itself, rather than creating instance data. Class-OWL (http://search.cpan.org/dist/Class-OWL/) intends to create Perl classes from OWL ontologies; however, the code appears to be non-functional, is not documented, and has not been updated for more than a year, so we assume the project has been abandoned.

Here, we present OWL2Perl—a Perl application that consumes OWL-DL ontologies and creates Perl packages representing each class in that ontology, in a manner similar to Jastor in Java (Kalyanpur *et al.*, 2004). Each Perl object includes 'stub' methods allowing you to get and set the properties that define that class. Importantly, instances of that Perl object can serialize themselves as ontologically valid instance data in RDF format.

2 DESCRIPTION

OWL2Perl is based on code from the open-source ODO project (Ontologies, Databases and Optimizations; Evanchik, 2006), a Perl framework for RDF manipulation. We have 'forked' and debugged the ODO codebase, as it does not appear to be actively maintained, and have renamed it PLUTO (available via CPAN under the Eclipse license). We have extended PLUTO to manage OWL ontologies as described below. OWL2Perl, then, is a set of modules and helper scripts that utilize the RDF/OWL parsing and model-building capabilities of PLUTO in order to generate object-oriented Perl modules from OWL classes.

Upon installing OWL2Perl, a Perl script 'owl2perl-generate-modules.pl' is placed in the distributions 'auto' path and can then be easily accessed by name. The script is run with a command-line

*To whom correspondence should be addressed.

argument indicating the URL of the OWL ontology you wish to represent as Perl modules. This ontology is retrieved and parsed, including imports of any ontology referred-to in that ontology. OWL2Perl then examines the property restrictions on each OWL Class, and creates a Perl module where each of those property restrictions becomes a gettable/settable facet of the module. That property is accessed by an object method named according to the predicate of that property restriction. The name of the module itself is derived from the URI of the class, thus ensuring there are no collisions when running OWL2Perl multiple times over different ontologies.

For example, assume we wish to create instances of the (simplistic) AnnotatedPDB OWL Class defined at:

```
http://sadiframework.org/ontologies/records.owl#AnnotatedPDB
```

The salient part of that ontology is the definition of the AnnotatedPDB class, which has the following OWL constraints:

```
<owl:Class rdf:about="http://sadiframework.org/ontologies/records.owl#AnnotatedPDB">
 <owl:equivalentClass>
  <owl:Class>
   <owl:intersectionOf rdf:parseType="Collection">
     <owl:Description rdf:about="http://purl.oclc.org/SADI/LSRN/PDB_Thing"/>
     <owl:Restriction>
       <owl:onProperty
rdf:resource="http://ontology.dumontierlab.com/hasReference"/>
          <owl:someValuesFrom
rdf:resource="http://purl.oclc.org/SADI/LSRN/PMID_Thing"/>
     </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
</owl:Class>
```

As shown, the class is defined as a PDB_Thing with a single property ‘hasReference’ that must have a PMID_Thing as its value. To generate the Perl module(s) representing this OWL class, we would execute the following command:

```
owl2perl-generate-modules.pl -i –u http://sadiframework.org/ontologies/records.owl
```

The ‘-i’ flag indicates that the script should follow owl-imports, and the ‘-u’ flag indicates that the OWL is to be retrieved from a URL, rather than a local file. Upon processing this ontology, a module is created with the following name:

```
sadiframework::org::ontologies::records::AnnotatedPDB
```

In addition, by following the import statement in the records.owl ontology, OWL2Perl generated another module called:

```
purl::oclc::org::SADI::LSRN::PMID_Thing
```

These modules are accessed and used in your Perl code as in the following example code snippet:

```
use lib ‘./Perl-OWL2Perl/generated’;
use sadiframework::org::ontologies::records::AnnotatedPDB;
use purl::oclc::org::SADI::LSRN::PMID_Thing;

# create the subject node
my $someURI=”http://pdb.org/12345”;
my $subject =
     sadiframework::org::ontologies::records::AnnotatedPDB->new($someURI);

# create the object node
my $object =
     purl::oclc::org::SADI::LSRN::PMID_Thing->new(”http://lsrn.org/PMID:163483”);

# link subject to object with the a predicate using a method created from
# the AnnotatedPDB’s OWL class definition
$subject->add_hasReference($object);

# serializing the OWL class
use OWL::Utils;
print STDOUT OWL::Utils::serialize($subject);
```

3 DISCUSSION

OWL2Perl was developed specifically to enhance Perl code support for the SADI (Semantic Automated Discovery and Integration) Semantic Web Service project; however, any project that must programmatically generate RDF data compliant with OWL-DL definitions could benefit from OWL2Perl. We believe the audience for OWL2Perl will expand as richer Description Logic OWL ontologies become increasingly pervasive in Bioinformatics.

ACKNOWLEDGEMENTS

We would like to acknowledge Stephen Evanchik for his work with the ODO project. We would also like to thank Martin Senger for his contribution to earlier versions of these code generation modules within the BioMoby project.

Funding: Genome Canada through the Genome Canada Bioinformatics Platform; CANARIE through funding of the C-BRASS Project of the NEP-2 competition. Core laboratory funding is derived from an NSERC Discovery grant.

Conflict of Interest: none declared.

REFERENCES

Antezana,E. *et al.* (2008) ONTO-PERL: an API for supporting the development and analysis of bio-ontologies. *Bioinformatics*, **24**, 885–887.

Baker,C.J.O. *et al.* (2008) Towards ontology-driven navigation of the lipid bibliosphere. *BMC Bioinformatics*, **9** (Suppl. 1), S5.

Evanchik, S. (2006) ODO – Ontologies, Databaes, and Optimizations a randon act of software.Available at http://stephen.evanchik.com/node/54 (last accessed date April 12, 2010).

Kalyanpur,A. *et al.* (2004) Automatic mapping of OWL ontologies into Java. In *Proceedings of the International Conference of Software Engineering and Knowledge Engineering (SEKE), June 20-24, 2004*, Banff, Canada.

Smith,B. *et al.* (2007) The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.*, **25**, 1251–1255.

Wolstencroft,K. *et al.* (2005) PhosphaBase: an ontology-driven database resource for protein phosphatases. *Proteins*, **58**, 290–294.