TEX-OWL: a Latex-Style Syntax for authoring OWL 2 ontologies

Matteo Matassoni[1], Marco Rospocher[2], Mauro Dragoni[2], and Paolo Bouquet[1]

[1] The University of Trento, Via Sommarive 9, Trento, I-38123, Italy
[2] Fondazione Bruno Kessler—IRST, Via Sommarive 18, Trento, I-38123, Italy

Abstract. This paper describes a new syntax that can be used to write OWL 2 ontologies. The syntax, which is known as TEX-OWL, was developed to address the need for an easy-to-read and easy-to-write plain text syntax. TEX-OWL is inspired by LATEX syntax, and covers all construct of OWL 2. We designed TEX-OWL to be less verbose than the other OWL syntaxes, and easy-to-use especially for quickly developing small-size ontologies with just a text editor. The important features of the syntax are discussed in this paper, and a reference implementation of a Java-based parser and writer is described.

1 Introduction and Motivation

Since OWL became a World (W3C) Wide Web Consortium recommendation, there has been a steady stream of Web (OWL) Ontology Language ontology editing tools that have made their way to users' desktops. Most notably, Protégé-OWL [1], Swoop [2], TopBraid Composer,[1] and MoKi [3].

All of these tools offer a variety of presentation or rendering formats for class, property and individual descriptions and axioms. These formats range from the W3C officially required RDF/XML exchange syntax [4], to the optionals: Turtle [5], OWL/XML [6], the Functional-Style Syntax [7], the Manchester Syntax [8], with some non-standard W3C syntaxes, like a Description-Logic style syntax and the Open (OBO) Biomedical Ontologies format [9].

While the use of ontology editing tools is becoming more and more popular, there are still situations where users have to quickly write small-size ontology for testing or prototyping purposes, and directly writing the ontology with a text editor would be more effective (i.e., the overhead of learning the ontology editing tool's functionalities and features is more than the benefit obtained by using it). These situations quite frequently occur in academic context.

W3C chose RDF/XML as the primary exchange syntax for OWL 2; indeed, this syntax must be supported by all OWL 2 tools. However, the fact that XML is extremely verbose and hard to write by hand excludes this syntax for quickly authoring and editing ontologies in a concise manner.

W3C provides alternatives to RDF/XML for OWL 2; these include Turtle, OWL/XML, the Functional-Style Syntax and the Manchester Syntax. OWL/XML is an XML

[1] `http://topbraidcomposer.com/`.

serialization for OWL 2 that mirrors its structural specification. It is more human-readable and easier to parse than RDF/XML; however like RDF/XML, OWL/XML is still XML. Another syntax that follows OWL 2's structural specification is the Functional-Style Syntax. It is a human-readable and plain text syntax that removes the burden of XML, however like the previous two, the Functional-Style Syntax is also verbose. In fact, it has an excessive number of keywords, and typically requires the use of a large number of brackets, as its name might suggest. The Manchester Syntax is a user-friendly syntax that can be used to write OWL 2 ontologies. It is the least verbose OWL 2 syntax and when it is used to write ontology documents, it gathers together information about names in a frame-like manner as opposed to the others OWL 2 syntaxes. This nature at a first look may seem a great advantage for the Manchester Syntax, but on the other hand it makes this syntax unable of handling General (GCI) Concept Inclusions (i.e., the Manchester Syntax does not cover the expressivity of the whole OWL 2 language).

The OWL Latex-Style Syntax was created to deal with the above issues and provide users with a lightweight syntax that makes it easier to write ontologies. It has been designed primarily for writing ontology documents in simple textual editor. The syntax is discussed in detail through the rest of this paper.

2 OWL Latex-Style Syntax

The full specification of the TEX-OWL syntax is available at `http://github.com/matax87/TexOwl/blob/master/docs/grammar.pdf`, together with several examples of using the various syntax constructs. The primary design consideration in developing TEX-OWL was to produce a syntax that was concise, and quick and easy to read and write by hand. We took inspiration for developing the syntax from the LATEX format, given it's popularity especially in academic environments. A previous attempt to develop a latex-like syntax was proposed in [11], but its syntax is restricted to a limited subset of OWL 1. Lessons learned from this previous experience were also taken into consideration. For example, keywords that represent datatypes and annotations (i.e., datatypes and annotations were hard-coded in the syntax) were remove in the new syntax to generalise them via IRIs.

It was also decided that although the syntax should be aligned as much as possible with the OWL specification, for example by using keywords derived from the Functional-Style Syntax specification, the main objective would be to strive for conciseness and a reduction in the amount of time it took users to write axioms. To this end, some new keywords were created and others changed in name or name length. Moreover, it was also decided that the syntax should match as much as possible the LATEX format peculiarities of using keyword and command that start with a backslash ('\') symbol, with required parameters inside curly braces and optional parameters inside square brackets.

Although the TEX-OWL Syntax borrows ideas from the OWL Functional-Style Syntax, it is much less verbose. An OWL ontology written in TEX-OWL starts with an optional preface and continues with the actual ontology document. The optional preface is where prefixes can be declared via the new keyword **\ns**. This keyword can be used also to declare a default prefix, which will be used for interpreting sim-

ple IRIs.[2] The actual ontology document begins with **begin**{**ontology**} and ends with **end**{**ontology**} syntax. After the begin ontology statement, users can also provide an optional ontology IRI and a even more optional version IRI typing them inside square brackets: **[ontologyIRI, versionIRI]**. Inside the begin/end block, user can import other ontology documents, using the keyword **import**, declare axioms and put ontology annotations, as shown in the example below:

```
\ns <http://www.mydomain.org/african#>
\begin{ontology}[<http://www.mydomain.org/african>]
  % Animals form a class
  animal \c
  % Plants form a class disjoint from animals
  animal \cdisjoint plant
  % Trees are a type of plant
  tree \cisa plant
  % Branches are parts of trees
  branch \cisa \oforall{is_part_of}{tree}
  % Leaves are parts of branches
  leaf \cisa \oforall{is_part_of}{branch}
  % Herbivores are exactly those animals that eat only plants or parts of plants
  herbivore \ceq (animal \cand \oforall{eats}{(plant \cor \oforall{is_part_of}{
      plant})})
  % Carnivores are exactly those animals that eat animals
  carnivore \ceq (animal \cand \oexists{eats}{animal})
  % Giraffes are herbivores, and they eat only leaves
  giraffe \cisa (herbivore \cand \oforall{eats}{leaf})
  % Lions are animals that eat only herbivores
  lion \cisa (animal \cand \oforall{eats}{herbivore})
  % Tasty plants are plants that are eaten both by herbivores and carnivores
  tasty_plant \cisa \candof{plant, \oexists{eaten_by}{herbivore}, \oexists{
      eaten_by}{carnivore}}
  eaten_by \oinv eats
  eats \odomain animal
\end{ontology}
```

3 Implementation

A Java based reference implementation of a TEX-OWL parser and writer were created.[3] They use the OWLAPI framework [10] and were developed as modules that can be integrated inside it. The parser was constructed using the Java (JavaCC) Compiler Compiler [12]. It can parse complete ontologies written in TEX-OWL. The writer, which inside the OWLAPI is known as *renderer*, can serialize OWLAPI's ontology objects to files written in TEX-OWL. Moreover, the implementation also includes converters, which can transform a TEX-OWL ontology to any other OWL 2 syntaxes and vice versa.

4 Concluding Remarks

TEX-OWL is a new OWL 2 syntax that was designed in response to a demand from users for a more concise syntax that can be easily used to quickly write small-size ontologies by hand. Key features of the syntax are that it is inspired by the LATEX syntax:

[2] Simple IRIs are equivalent to abbreviated IRIs where the default prefix is used and there is no need need of typing the colon (':') symbol.

[3] The implementation is available from `http://github.com/matax87/TexOwl/`.

in particular the syntax uses the same format for parameters and keywords. The syntax is suited for use in simple textual editor tools. A reference implementation of a Java based parser and a writer have been produced, which may be integrated into any tool. The implementation also includes converters, which can transform TEX-OWL to other OWL 2 syntaxes and vice versa.

In order to evaluate TEX-OWL, two questionnaires were designed and sent to knowledge engineers with experience in authoring ontologies with the various OWL syntaxes. In the first questionnaire (accessible here: `http://goo.gl/Cjpqtg`), all OWL 2 syntaxes and TEX-OWL's intuitiveness, conciseness, and understandability were compared using ten different examples of use. The second questionnaire (accessible here: `http://goo.gl/lbFu4R`) was focused on evaluating the usability of the new syntax for authoring a small ontology. Ten knowledge engineers participated to the first questionnaire, and five of them took part to the second one. Ratings were expressed according to the typical five-level Likert scale.

In summary, the results show that TEX-OWL is indeed the most concise syntax and is intuitive as the Manchester Syntax, which is the most intuitive among OWL 2 syntaxes. Moreover, users have found it easy to use TEX-OWL for authoring a small example ontology and that, in general, this syntax is better to use for writing ontologies by hands than other OWL 2 syntaxes.

References

1. Knublauch, H., Musen, M.A., Rector, A.L.: Editing description logics ontologies with the Protégé OWL plugin (2004)
2. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies (2005)
3. Chiara Ghidini, Marco Rospocher, Luciano Serafini: Modeling in a Wiki with MoKi: Reference Architecture, Implementation, and Usages International Journal On Advances in Life Sciences, IARIA, volume 4, 111-124 (2012)
4. Fabien, G., Schreiber, G.: Rdf 1.1 xml syntax specification (2014) `http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/`.
5. Beckett, D.: New syntaxes for rdf. Technical report, Institute For Learning And Research Technology, Bristol (2004)
6. Motik, B., Parsia, B., Patel-Schneider, P.F.: Owl 2 web ontology language xml serialization (second edition) (2012) `http://www.w3.org/TR/2012/REC-owl2-xml-serialization-20121211/`.
7. Motik, B., Parsia, B.: Owl 2 web ontology language structural specification and functional-style syntax (second edition) (2012) `http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/`.
8. Horridge, M., Drummond, N., Goodwin, J., Rector, A.L., Stevens, R., Wang, H.: The manchester owl Syntax (2006)
9. Motik, B., Parsia, B.: Obo flat file format 1.4 syntax and semantics [draft] (2011) `ftp://ftp.geneontology.org/go/www/obo-syntax.html`.
10. The owl api `http://owlapi.sourceforge.net`.
11. Latex2owl, `http://dkm.fbk.eu/index.php/Latex2owl`.
12. Sreeni, V., Sriram, S.: Java compiler compiler [tm] (javacc [tm]) - the java parser generator `http://javacc.java.net`.