Authoring OWL 2 ontologies with the TEX-OWL syntax

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 [3], Swoop [4], TopBraid Composer,[1] and MoKi [5].

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 [6], to the optionals: Turtle [7], OWL/XML [8], the Functional-Style Syntax [9], the Manchester Syntax [10], with some non-standard W3C syntaxes, like a Description-Logic style syntax and the Open (OBO) Biomedical Ontologies format [11].

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 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

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

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 [1], 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 [13], but its syntax and tools are restricted to a limited subset of OWL 1. Lessons learnt 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. The ontology in Listing 1.1, the TEX-OWL version of the example in [2, p. 129], illustrates some of the main constructs of the syntax, while we refer the reader interested in the syntax of all OWL 2 constructs to the complete specification in [1]. 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 simple 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. To favour the readability of the ontology example in Listing 1.1, each TEX-OWL statement is preceded by its natural language phrasing, injected in the ontology code as TEX-OWL comments ('%'), and an indication of the type of construct used.

Listing 1.1. An African Wildlife Ontology in TEX-OWL

```
\ns <http://www.mydomain.org/african#>
\begin{ontology}[<http://www.mydomain.org/african>]
  % Animals form a class [Class declaration]
  animal \c
  % Plants form a class disjoint from animals [Disjoint classes]
  animal \cdisjoint plant
  % Trees are a type of plant [Subclass Axiom]
  tree \cisa plant
  % Branches are parts of trees [Subclass Axiom, Object Property
       Universal Quantification]
  branch \cisa \oforall{is_part_of}{tree}
  % Leaves are parts of branches [Subclass Axiom, Object
      Property Universal Quantification]
  leaf \cisa \oforall{is_part_of}{branch}
  % Herbivores are exactly those animals that eat only plants or
       parts of plants [Class Equivalence, Class Expression,
      Object Property Universal Quantification]
  herbivore \ceq (animal \cand \oforall{eats}{(plant \cor
      \oforall{is_part_of}{plant})})
  % Carnivores are exactly those animals that eat animals [Class
       Equivalence, Class Expression, Object Property
      Existential Quantification]
  carnivore \ceq (animal \cand \oexists{eats}{animal})
  % Giraffes are herbivores, and they eat only leaves [Subclass
      Axiom, Class Expression, Object Property Universal
      Quantification]
  giraffe \cisa (herbivore \cand \oforall{eats}{leaf})
  % Lions are animals that eat only herbivores [Subclass Axiom,
      Class Expression, Object Property Universal Quantification
      ]
  lion \cisa (animal \cand \oforall{eats}{herbivore})
  % Tasty plants are plants that are eaten both by herbivores
      and carnivores [Subclass Axiom, Sequence Intersection of
      Class Expressions, Object Property Universal
      Quantification]
```

[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.

```
  tasty_plant \cisa \candof{plant ,\oexists{eaten_by}{herbivore
      },\oexists{eaten_by}{carnivore}}
  % eats and eaten_by are inverse of each other [Inverse Object
      Property]
  eaten_by \oinv eats
  % Everything that eats is an animal [Object Property Domain]
  eats \odomain animal
\end{ontology}
```

3 A TEX-OWL parser and writer

A Java based reference implementation of a TEX-OWL parser and writer were created.[3] They use the OWLAPI framework [12] and were developed as modules that can be integrated inside it. The parser was constructed using the Java (JavaCC) Compiler Compiler [14]. 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 Syntax Evaluation

In this Section, we present the evaluation performed with the help of ontology experts in order to analyze if the presented syntax is suitable, easy-to-use, and comprehensive enough for ontology authoring.

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,[4] all OWL 2 syntaxes and TEX-OWL's intuitiveness, conciseness, and understandability were compared using ten different examples of use. In details, each example of use consists in a set of axioms expressed by using the 6 syntaxes that we compared, namely, TEX-OWL, Manchester, Turtle, Functional, RDF/XML, and OWL/XML.

Table 1 presents the results extracted from the survey submitted to the ontology experts. Ten knowledge engineers participated to this questionnaire. Each value of the table represents, for each syntax, how many experts, averaged over the whole set of example, judged the syntax intuitive and concise with respect to the proposed examples.

We may observe that, in general, there is a clear distinction between the first three syntaxes (TEX-OWL, Manchester, and Functional) with respect to the other three (Turtle, OWL/XML, and RDF/XML). Such a distinction is clearly visible from the judges about their intuitiveness and their conciseness.

By comparing, directly, the TEX-OWL syntax with the Manchester and the Functional ones, we may notice that the intuitiveness of the TEX-OWL syntax is comparable with the Manchester one, but the TEX-OWL syntax obtained a significant better score

[3] The implementation is available from `http://github.com/matax87/TexOwl/`.
[4] Accessible here: `http://goo.gl/Cjpqtg`

Table 1. Results obtained on the survey submitted to the experts concerning the comparison between the most important ontology authoring syntaxes.

Syntax Name	Intuitiveness	Conciseness
LaTeX-like	6.5	9.7
Manchester	6.8	2.7
Functional	4.8	5.3
Turtle	1.1	1.0
OWL/XML	1.3	0.1
RDF/XML	0.4	0.0

for conciseness with respect to the Manchester syntax. Moreover, we recall that the Manchester syntax does not cover the whole OWL 2 language: it does not properly support the expression of General Concept Inclusions (GCIs) (such as "having a parent who is a person, implies being a person"), which instead are expressible in TEX-OWL (resp., "`\oexists{hasParent}{Person} \cisa Person`").

The only syntax that has been partially judged concise is the Functional one; however, the difference between the scores obtained with respect to the TEX-OWL syntax is different enough to state that, from the conciseness point of view, the TEX-OWL syntax is the most suitable solution for the ontology authoring task.

The second questionnaire[5] was focused on evaluating the usability of the new syntax for authoring a small ontology. The task to be completed consists in developing, by starting from a textual description, the ontology shown in Listing 1.1. Experts were asked to judge how much the developing of a sample ontology was easy and intuitive, and to rate the simpleness of developing the sample ontology with the LaTeX-like syntax with respect to syntaxes previously used by the experts. Ratings were expressed according to the typical five-level Likert scale. More in details, experts were asked to answer to three questions about the TEX-OWL syntax: if it was difficult to develop the ontology (on a scale from 1 to 5, where 5 mean "Very easy"); if the TEX-OWL syntax was easy to remember (on a scale from 1 to 5, where 5 means "Hard to remember"); and, by comparing with their previous authoring experiences with other syntaxes, if the use of the TEX-OWL syntax was better (on a scale from 1 to 5, where 5 means "Definitely better").

Six knowledge engineers took part to this evaluation. On average, the TEX-OWL syntax obtained a score of 3.5 on the first question, 3.17 on the second questions, and 3.67 on the third one. Therefore, we may state that the experts rated the task of developing an ontology in TEX-OWL more easy than difficult, that the syntax is slightly more easy than difficult to remember, and that the use of TEX-OWL syntax was quite better than previous authoring experiences with other syntaxes.

5 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 on-

[5] Accessible here: `http://goo.gl/lbFu4R`

tologies by hand. A characterizing feature of the syntax is that it is inspired by the LaTeX syntax: in particular the syntax uses the same format for parameters and keywords as used in LaTeX. The syntax is suited for use in simple textual editor tools. A reference implementation of a Java based parser and writer has 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.

The evaluation results shown that TeX-OWL is the most concise syntax and has an intuitiveness comparable with 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. TeX-OWL Syntax Grammar: `http://github.com/matax87/TexOwl/blob/master/docs/grammar.pdf`
2. Antoniou, G., and van Harmelen, F.: A Semantic Web Primer, The MIT Press, (2004)
3. Knublauch, H., Musen, M.A., Rector, A.L.: Editing description logics ontologies with the Protégé OWL plugin (2004)
4. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies (2005)
5. 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)
6. Fabien, G., Schreiber, G.: Rdf 1.1 xml syntax specification (2014) `http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/`.
7. Beckett, D.: New syntaxes for rdf. Technical report, Institute For Learning And Research Technology, Bristol (2004)
8. 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/`.
9. 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/`.
10. Horridge, M., Drummond, N., Goodwin, J., Rector, A.L., Stevens, R., Wang, H.: The manchester owl Syntax (2006)
11. Motik, B., Parsia, B.: Obo flat file format 1.4 syntax and semantics [draft] (2011) `ftp://ftp.geneontology.org/go/www/obo-syntax.html`.
12. The owl api, `http://owlapi.sourceforge.net`.
13. Latex2owl, `http://dkm.fbk.eu/index.php/Latex2owl`.
14. Sreeni, V., Sriram, S.: Java compiler compiler [tm] (javacc [tm]) - the java parser generator `http://javacc.java.net`.