Rendering OWL in Description Logic Syntax

Cogan Shimizu[1], Pascal Hitzler[1], and Matthew Horridge[2]

[1] Data Semantics (DaSe) Laboratory, Wright State University, Dayton, OH, USA
[2] Bio-Medical Informatics Research Group, Stanford University, Stanford, CA, USA

Abstract. As ontology engineering is inherently a multidisciplinary process, it is necessary to utilize multiple vehicles to present an ontology to a user. In order to examine the formal logical content, description logic renderings of the axioms appear to be a very helpful approach for some. This paper introduces a number of changes made to the OWLAPI's LaTeX rendering framework in order to improve the readability, concision, and correctness of translated OWL files, as well as increase the number of renderable OWL files.

1 Motivation

For ontology developers and consumers intimately familiar with the logical and formal semantic underpinnings of OWL, the presentation of OWL files in the form of description logic syntax appears to be a very useful one for a quick assessment of expressivity and formal content.

The OWLAPI [1], which is a powerful tool for the programmatic construction, manipulation, and rendering of ontologies, has for considerable time had limited support for the rendering of OWL ontologies in description logic syntax via LaTeX. Unfortunately, this LaTeX rendering framework, which outputs description logic in a LaTeX source file, was never developed beyond an early experimental stage. As a consequence, translations suffered from a number of syntax errors and poor readbility of the output. In practice, translations were further impacted by the presence of illegal characters in the LaTeX source, thus preventing nearly all renderings from typesetting. In Section 3 we see that in a test set of 117 OWL files, not a single one did typeset without error. This paper addresses changes made to the OWLAPI LaTeX rendering framework in order to improve translations' succinctness, readability, and syntax, as well as ensuring that a larger number of translations will indeed typeset.

In Section 2 we describe in more detail the changes we made to the OWLAPI. Note that no changes were made to the rendering behaviour of SWRL or annotations. In Section 3 we describe the tools we developed, our test set, and rendering results.

2 Improvements

For context, we provide a very brief overview of how the OWLAPI renders an ontology in LaTeX. First, the renderer examines a loaded ontology. Then, for each

entity, (i.e Class, Object Property, Data Property, Individual, and Datatype) in the ontology it prints associated axioms and facts. An axiom is associated to an entity if it appears somewhere in the axiom. For example, the axiom

DisjointClasses(A, B, C)

is associated with classes A, B, and C. While this does result in redundantly rendered axioms, we stress that the renderer is meant to summarize the *entities* in an ontology, rather than exhaustively enumerate all axioms in the ontology. Below, we describe the main changes made to enhance the framework's ability to do so.

Datatypes With respect to the syntax of datatypes, there were a number of subtle changes necessary to align the LaTeX renderer with the OWL standard [3]. These changes are doubly important in that they prevent the writing of illegal characters (e.g. '#') and increase the readability of the rendering. For datatypes that are defined in the current namespace, their namespaces are omitted. Externally defined datatypes' namespaces are included using shortform notation. For example, datatypes specified as XML Schema Datatypes or in RDFS are prepended with the popular, shortened namespaces of xsd and rdfs, respectively.

Nominals Literals, when used as nominals, are now properly rendered using set notation. In accordance with the above, the example below includes a shortform namespace for its datatype.

∃hasSigrid3IceFormCode.{"05"^^xsd:string}

DatatypeRestriction Axiom Previously, DatatypeRestriction axioms were not rendered in an intuitive manner. We have made changes in order to make it more similar to the functional syntax specified in [3]. However, we diverge slightly from the specification in the interest of readability. The constrained datatype is followed by a colon to differentiate it from its facets. Further, the constraining facets are rendered using their respective relational operators instead of keywords. In general, DatatypeRestriction axioms are now rendered using the following form, where the '+' indicates one or more of the preceding tokens.

DatatypeRestriction(datatype: (constrainingFacet restrictionValue)+)

HasKey Axiom The HasKey axiom has no analog in description logic [2]. We also contend that the functional syntax in [3] is unwieldy and that distinguishing between Object Properties and Data Properties is unnecessary for axiomatic rendering. As such, we have adopted the following syntax for a HasKey axiom, where the '+' means one or more of the preceding token.

ClassExpression *hasKey* (Property+)

Miscellaneous Fixes

- The Subproperty axiom now properly renders the subproperty.
- Extraneous spacing after logical symbols (e.g. $\neg$) has been fixed.
- Axioms expressing cardinality now correctly render cardinality.
- Role restriction axioms now have correct "." syntax.

Spacing & Math Mode We have also made several general changes to increase both the quality of the LaTeX source and readability of the rendering itself. In particular, the amsmath package is now included in the preamble so that we may align related axioms over their principal relation (i.e. $\equiv$, $\not\equiv$, $\sqsubseteq$) or after a function name. As such, axioms are now rendered in math mode.

Line Breaking Heuristics In some cases, axioms would result in an excessively long rendering (i.e. result in hbox overflow, placing text in or even beyond the page margin). For the most part, LaTeX handles itself in knowing when to break a line. However, this behavior does not occur in the math environments. As such, it was necessary to look into methods for preventing unacceptable overflow.

The first option examined was the LaTeX package, breqn. This package is an experimental package that employs its own heuristics for breaking excessively long equations. Unfortunately, breqn's heuristics take into account only a select number of operators as potential breaking points. Due to the uncommon operators that description logic employs, breqn was unable to find appropriate breaking points.

The next option was the split environment from the LaTeX package amsmath. However, split does not dynamically split an equation; it is an entirely manual process. At this point, we developed our own heuristics to determine when the split environment would be necessary.

In the rendering tool, we have introduced a middle layer to the rendering system. The OWLAPI LaTeX framework renders normally, but to a special temporary file. From this temporary file, we examine the LaTeX source code. For our test set, this approach did not result in significant additional runtime. The heuristics is defined as follows.

First, we control for the LaTeX commands that are employed by the rendering framework and then count an empirically determined number of characters; we found 125 characters to be a reasonable equation length before a newline would be required. The following is an example rendering following this heuristics.

$$\begin{aligned}\mathrm{DataGranule}(x_1) \rightarrow \geq 1 x_2\ \mathrm{hasDataSet}(x_1, x_2) \wedge \mathrm{DataSet}(x_2) \\ \wedge \leq 1 x_3\ \mathrm{hasDataSet}(x_1, x_3) \wedge \mathrm{DataSet}(x_3)\end{aligned}$$

There are some limitations to this approach, as each entity's subsection is a single align environment. The split environment is, in turn, embedded in it. As such, if the antecedent of an axiom is very long, the line breaks may occur in or beyond the margin.

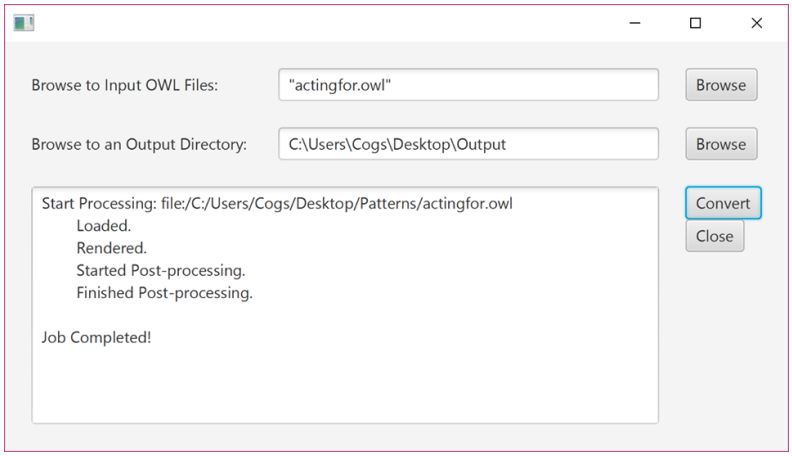

Fig. 1. Snapshot of the GUI tool.

Reduction of Duplicate Axioms Several OWL concepts provide a way for succinctly expressing pairwise relations (e.g. equivalence and disjointness). However, the translations of these concepts into description logic can potentially generate a huge number of axioms. For example, in order to express that n classes are mutually disjoint requires $2 \cdot \binom{n}{2}$ axioms. Furthermore, under the current framework all these axioms are related and will thus be printed in each class's section, for a total of $2n \cdot \binom{n}{2}$ axioms. This can quickly obscure the actual relationship between all the classes. As such, we adopt the functional syntax as defined in the specification as follows

$$\mathit{disjoint}(c_1, c_2, c_3, \cdots, c_n)$$

3 Results

All tools, source code, the test set, and rendering results are available for download from the Data Semantics Lab website.[3]

Tools In order to make these changes to the LaTeX renderer accessible, we have developed GUI and CLI interfaces. Fig. 1 shows a capture of the developed GUI tool. The tool can take any number of files located in a single directory and output LaTeX source files into a user specified directory. A small log window is provided for monitoring job progress.

In addition, the changes described in this paper (and those used in the developed tools) have been submitted to the OWLAPI maintainers for review. At

[3] `http://daselab.org/content/owl2dl-rendering`

the time of this writing, the changes are visible on the GitHub repository and will appear in the version 5.0.6 release.

Test Set & Rendering Results In order to test our changes, we pulled ontology design patterns from the `www.ontologydesignpatterns.org` website. In total, we collected 117 OWL files. These represent the subset of all Ontology Design Patterns from this site that are well-formed, syntactically correct, and have active download links. We chose to use Ontology Design Patterns as our test set, as they are ideal use cases for the rendering framework. That is, examining the logical structure of a module is an important step in ontology engineering.

First, we note that prior to our changes to the LaTeX renderer, none of the 117 OWL files would typeset without error due to illegal characters present in the expanded namespaces of the datatypes. Additionally, 2 of the 117 files generated lines in excess of the margins of the page when rendering was forced.

After translating all 117 files using the GUI tool, all LaTeX source files typeset without error and without needing manual modification. Further, the heuristic line breaking accurately and reasonably breaks the excessively long axioms found previously.

Future Work The ontology engineering process necessarily includes domain experts. These domain experts are not expected to be experts in logic or OWL. We view this tool (and the changes to the OWLAPI) as a necessary step in providing multiple ways for domain experts to interface with OWL. Future work will consider adapting the LaTeX rendering framework and the lessons herein learned to other logical syntaxes. Furthermore, as these changes have been submitted to the OWLAPI, this is a perfect springboard to make the LaTeX rendering available via a plug-in to Protégé.

Demonstration For the demonstration, we will provide a brief tutorial on acquiring the tool and its usage. Then, we will demonstrate its functionality via live renderings of some ontologies. Furthermore, we will invite users to provide their own ontology to view performance on ontologies outside of our testset.

Acknowledgement. The first two authors acknowledge support by the National Science Foundation award 1440202 EarthCube Building Blocks: Collaborative Proposal: GeoLink – Leveraging Semantics and Linked Data for Data Sharing and Discovery in the Geosciences.

References

1. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21, 2011.
2. M. Krötzsch, F. Simančík, and I. Horrocks. A description logic primer. In J. Lehmann and J. Völker, editors, *Perspectives on Ontology Learning*, chapter 1. IOS Press, 2014.
3. B. Motik, P. Patel-Schneider, and B. Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation 11 December 2012, 2012.