The Manchester OWL Syntax

Matthew Horridge[1], Nick Drummond[1], John Goodwin[2], Alan Rector[1], Robert Stevens[1], and Hai H Wang[1]

[1] The University of Manchester
[2] The Ordnance Survey

Abstract. This paper describes a new syntax that can be used to write OWL ontologies, and fragments of OWL ontologies for presentation and editing purposes. The syntax, which is known as the *Manchester OWL Syntax*, was developed in response to a demand from a wide range of users, who do not have a Description Logic background, for a "less logician like" syntax. The Manchester OWL Syntax is derived from the OWL Abstract Syntax, but is less verbose and minimises the use of brackets. This means that it is quick and easy to read and write. The important features of the syntax are discussed, and a reference implementation of a Java based parser is described.[3]

1 Introduction and Motivation

Since OWL became a W3C recommendation, there has been a steady stream of OWL ontology editing tools that have made their way to users' desktops. Most notably, Protégé-OWL [1], from Stanford Medical Informatics, and Swoop [2] from the Mindswap lab at the University of Maryland.

Both of these tools offer a variety of presentations or renderings for class, property and individual descriptions and axioms. These presentations range from the officially recommended RDF/XML exchange syntax [3], to a Description Logic style syntax, with Turtle/N-Triples [6], and the OWL Abstract Sytnax [4] somewhere between the ends of this syntax spectrum.

Experience of delivering several OWL tutorials and workshops, mainly for domain experts, including bio-informaticians, geographers and e-scientists, has made it evident that for the vast majority of non-logicians, none of the above syntaxes are suitable for writing class expressions and other types of axioms. In particular, the existing OWL syntaxes are either too verbose, or too complicated.

1.1 W3C OWL Syntaxes

A quick read through of the W3C OWL web pages leaves no room for doubt that the preferred OWL syntax is RDF/XML. Even the OWL guide uses this syntax for the presentation of examples. However, the verbosity of the XML,

[3] It should be noted that the description given in this paper is informative, for a normative specification see http://www.co-ode.org.

and the fact that it is difficult to write by hand, rule this syntax out for quickly writing and editing class descriptions in a concise manner.

An alternative to the RDF/XML syntax is the OWL Abstract Syntax [4]. This syntax is a high level, human readable OWL syntax. The Abstract Syntax is frequently used to exchange snippets of OWL via e-mail messages and discussion lists. However, like the RDF/XML syntax, the Abstract Syntax is also verbose – it has an excessive number of keywords, and typically requires the use of a large number of brackets.

The Semantic Web Best Practices Working Group [5] settled on Turtle – an N3 derivative [6], for posting snippets of ontologies in e-mail discussions and for presentation in their best practice documents. Turtle is a triple based notation, and, amongst others, is favoured by Tim Berners Lee. It was primarily designed to represent RDF. Hence, representation of OWL class descriptions and other constructs in Turtle exposes the triples that are used to encode ontology constructs. It is arguable that when written in Turtle, the meaning of OWL entities is obfuscated because of the representation of raw triples.

1.2 The German DL Syntax and Protégé-OWL Compact Syntax

For the presentation of class descriptions and class axioms, both Protégé-OWL and Swoop defaulted to a syntax that is favoured by the logicians – the German DL Syntax. This syntax uses description logic symbols such as $\exists$, $\forall$, $\sqcap$, $\neg$. [4] Examples from Protégé-OWL and Swoop are shown in Figure 1 and Figure 2 respectively.

Given the Description Logic underpinnings of OWL, and the compactness of the DL Syntax, it perhaps unsurprising that this was the syntax of choice for the major OWL tools. However, it has been evident that the DL syntax isn't the preferred syntax for non-logicians. Indeed, the German DL Syntax presents an extra hurdle for non-logicians when learning OWL. It has been observed that domain experts, who do not have a DL background, recoil at the sight of backwards Es and upside down As. They find the DL style syntax both difficult to read and write.

Coupled with the problem of containing cryptic symbols, the syntax for restrictions is a prefix syntax. That is, the restriction quantifier precedes the role/property name and optional filler. It has has been observed that this can lead users to initially read restrictions incorrectly. For example, many users intially read, $\exists$ hasTopping MozzarellaTopping as, "some pizzas have toppings that are mozzarella topping", compared with the correct reading, "all pizzas have toppings that are some mozzarella topping".

Additionally, due to the special symbols required by the DL syntax, it is difficult to paste snippets of ontologies into e-mails, discussion forums and presentation slides, meaning that it isn't an ideal human readable exchange syntax.

[4] The default syntax for Protégé-OWL was actually a syntax that was derived from the German DL Syntax, called the *Compact OWL Syntax*. This syntax was a confusing mixture of prefix and infix notation.

1.3 Summary

The Manchester Syntax was created to deal with the above issues and provide non-logicians with a syntax that makes it easier to write ontologies. It has been designed primarily for presenting and editing class expressions in tools, but it can also be used for representing complete ontologies. The syntax is discussed in detail through the rest of this paper.

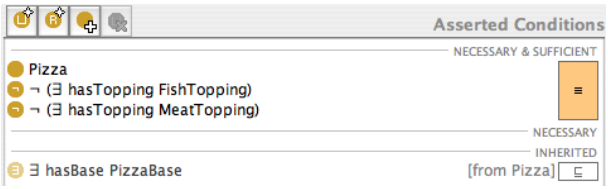

Fig. 1. An Example of the original DL syntax used in Protégé-OWL . The figure shows the description of a VegetarianPizza as defined in the pizza ontology.

Fig. 2. An Example of the syntax used in Swoop . The figure shows the description of a VegetarianPizza as defined in the pizza ontology.

2 The Manchester OWL Syntax

2.1 Design Considerations

The primary design considerations were to produce a syntax that was concise, did not use DL symbols, and was quick and easy to read and write. These considerations were based on the experience and interaction with users of Protégé-OWL where the syntax would be primarily used to edit class expressions. Lessons learnt from the GALEN project [7] were also taken into consideration. For example, minimsing the number of brackets required to write class expressions, and choosing keywords to promote readability, were taken into account.

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 OWL abstract syntax specification, the main objective would be to strive

for readability and a reduction in the amount of time it took domain experts and non-logicians to understand the information that was being represented. To this end, several new keywords were created.

2.2 The Syntax

Although the Manchester OWL syntax borrows ideas from the OWL Abstract Syntax, it is much less verbose. Whilst following the compactness of the German DL syntax, special mathematical symbols such as $\exists$ $\forall$, $\neg$ and have been replaced by more intuitive keywords such as `some`, `only`, and `not`.

A significant design decision was to use an infix notation rather than a prefix notation for keywords that are used in restrictions. This was decision was made in order to directly combat the problem of non-logicians misreading class expressions as described in Section 1.2.

2.3 Class Descriptions

One of the main driving factors behind the Manchester OWL Syntax was to produce a syntax that could be used to edit class descriptions in tools such as Protégé-OWL or Swoop . The class description syntax is shown in Figure 3. Keyword symbols are shown in bold caps, however, capitalisation of keywords is optional – for on screen presentations, it has been found that lower case keywords with colouring and syntax highlighting work particularly well.

In addition to making class expressions more natural to read, the natural language keywords, also makes it easy to paste the plain text representation of the expression into e-mails etc. without incurring the formatting problems that can arise due to the different fonts required to represent the mathematical symbols that are used in the DL syntax.

OWL Constructor	DL Syntax	Manchester OWL S.	Example
intersectionOf	C $\sqcap$ D	C **AND** D	Human **AND** Male
unionOf	C $\sqcup$ D	C **OR** D	Man **OR** Woman
complementOf	$\neg$ C	**NOT** C	**NOT** Male
oneOf	{a} $\sqcup$ {b}...	{a b ...}	{England Italy Spain}
someValuesFrom	$\exists$ R C	R **SOME** C	hasColleague **SOME** Professor
allValuesFrom	$\forall$ R C	R **ONLY** C	hasColleague **ONLY** Professor
minCardinality	$\geq$ N R	R **MIN** 3	hasColleague **MIN** 3
maxCardinality	$\leq$ N R	R **MAX** 3	hasColleague **MAX** 3
cardinality	= N R	R **EXACTLY** 3	hasColleague **EXACTLY** 3
hasValue	$\exists$ R {a}	R **VALUE** a	hasColleague **VALUE** Matthew

Fig. 3. The Manchester OWL Syntax OWL 1.0 Class Constructors

Precedence The Manchester OWL Syntax encourages the minimisation of the number of brackets that are used. This is achieved using operator precedence for class descriptions. The following list summarises operator precedence – operators are shown from highest precedece to lowest precedence.

- SOME, ALL, VALUE, MIN, MAX, EXACTLY, THAT
- NOT
- AND
- OR

As would be expected, the syntax supports the nesting of class constructors to arbitrarily complex levels. Complex class expressions can be disambiguated by bracketing. For example, the class expression below describes the set of people who have at least one child that has some children that are only men (i.e. grandparents that only have grandsons). The expression has been formatted using indentation to aid readability.

```
Person AND
hasChild SOME (Person AND
                    (hasChild ONLY Man) AND
                    (hasChild SOME Person))
```

The 'THAT' Keyword The "THAT" keyword was introduced into the sytax to make certain class expressions read more naturally. The inspiration for this was taken from the syntax that was developed as part of the GALEN project [7]. The keyword "THAT" is in fact a synonym for "AND". It is used after named classes which precede restrictions. In the example below, the previous example has been rewritten to use the THAT keyword – it is noticable that the expression is more readable.

```
Person THAT
hasChild SOME (Person THAT
                    (hasChild ONLY Man) AND
                    (hasChild SOME Person))
```

2.4 OWL Entity Descriptions

Class expressions that are built up using the syntax described previously can be used in tools for presenting and editing items such as superclass/equivalent class expressions etc. Figure 4 shows an example of the Manchester OWL Syntax being used to represent the concept of a VegetarianPizza in Protégé-OWL .

In addition to the class expression syntax, there is a full syntax for OWL entity descriptions. This means that it is possible to represent full descriptions for classes, properties and individuals in a textual manner. An example of the syntax for named class descriptions is given in Figure 5 and an EBNF style grammar shown in Figure 6. Such textual descriptions are ideal for use in non-DL

Fig. 4. An example of the Manchester OWL Syntax being used to represent the concept of a VegetarianPizza in Protégé-OWL

papers, or in e-mail discussion lists and forums. As can be seen from Figure 5, it is possible to represent annotations on classes and class axioms using the syntax. Inspiration for this annotation syntax was taken from the JavaDoc [8] syntax that is used for documenting Java classes and methods.

For the sake of brevity, the full class description syntax and grammar for properties and individuals is not shown in this paper – full specifications can be found on the CO-ODE website[5]. However, property and individual descriptions follow the same style as class descriptions.

```
/**
 * @rdfs:comment A vegetarian pizza is a pizza that only has cheese toppings
 *               and tomato toppings.
 *
 * @rdfs:label Pizza [en]
 * @rdfs:label Pizza [pt]
 */
Class: VegetarianPizza

EquivalentTo:

    Pizza and
    not (hasTopping some FishTopping) and
    not (hasTopping some MeatTopping)

DisjointWith:

    NonVegetarianPizza
```

Fig. 5. An example of the Manchester OWL Syntax being used to represent a full class description.

[5] http://www.co-ode.org

[**Annotation**]
'Class:' **classID** {**Annotation**
(('SubClassOf:' **ClassExpression**)
| ('EquivalentTo' **ClassExpression**)
| ('DisjointWith' **ClassExpression**)) }

ClassExpression = A class expression that is constructed using the class constructors shown in Figure 3.

Fig. 6. EBNF for OWL Class Descriptions

3 Design Patterns and Macros

3.1 ONLYSOME

A common ontology desing pattern is to combine a set of existential restrictions that act along a given property with a universal restriction that acts along the same property and has a filler that is the union of the existential fillers. The universal restriction is sometimes known as a *closure axiom*. The following example is taken from the well known pizza ontology[6].

Pizza THAT
hasTopping SOME MozzarellaTopping AND
hasTopping SOME TomatoTopping AND
hasTopping SOME PeperroniTopping AND
hasTopping ONLY (MozzarellaTopping OR
TomatoTopping OR
PepperonniTopping)

Since this is a common pattern, the Manchester OWL Syntax provides a shortcut macro. The macro takes the form, R ONLYSOME [C_0,... C_n], which expands to,

R SOME C_0 AND
⋮
R SOME C_n AND
R ONLY (C_0 OR ... OR C_n)

Figure 7 shows the above example expressed using the onlysome macro – the reduction in verbosity should be evident.

Similar macros, using the SOME and ONLY keywords, are also available for sets of existential restrictions and sets of universal restrictions respectively. The SOME keyword macro expands to an existential restriction for each class listed between square brackets. The ONLY keyword macro expands to a single

[6] http://www.co-ode.org/ontologies/pizza

restriction that has a filler, which is the disjunction of the classes listed between square brackets. Hence, both macros encourage neophyte users to "do the right thing".

```
Pizza THAT
hasTopping ONLYSOME [ MozzarellaTopping,
                      TomatoTopping,
                      PepperonniTopping]
```

Fig. 7. An example of the **ONLYSOME** macro

3.2 Value Partitions

The Semantic Web Best Practices Working Group [5] have published several ontology desgin patterns. The Manchester OWL Syntax includes a shortcut macro for creating one of these commonly used patterns – *Value Partitions*. This syntax for value partitions is:

'ValuePartition:' **className**
objectPropertyID ['**classID classID** {**classID**}']'

This expands to produce a functional property, and a list of disjoint classes which cover the value partion class that is identified by 'className'.

3.3 Exclusive OR

Ontology development at the Ordnance Survey highlighted a need for a compact way of representing an exclusive OR. In other words, A OR B but not both A and B. In line with some programming languages, the exclusive OR macro was introduced. The macro simply uses the XOR keyword. For example C XOR D, which expands to (C OR D) AND NOT (C AND D).

4 Implementation

A Java based reference implementation of a Manchester OWL Syntax parser was created.[7] The parser can parse class expressions, class, property and individual descriptions, and complete ontologies written in the Manchester OWL Syntax. It was constructed using the Java Compiler Compiler (JavaCC) [9]. A version of the parser that creates class expressions or descriptions, and ontologies using the WonderWeb OWL API [10] is also available.

[7] The implementation is available from http://www.co-ode.org

5 OWL 1.1 Support

The Manchester OWL Syntax was developed prior to the OWL 1.1 specification. However, the syntax has been extended, with minimal effort, to support OWL 1.1. Such extensions included support for QCRs, ValueNot, Disjoint unions, user derived datatypes and the like. Full specifiactions can be found on the CO-ODE web site.

6 Informal Evaluation

Since the Manchester OWL Syntax was created, it has become the default syntax in Protégé-OWL . Many non-logician users, such as members of the BioPAX consortium[8], the Ontogenisis Network and the Ordnance Survey[9] have commented that they much prefer the syntax to the previous DL style syntax. They find it easier to grasp and in some cases, such as at the Ordnance Survey, it has lowered the barrier to being able to read and interpret ontologies[10].

The Manchester OWL Syntax has also had commercial success. The recently released OWL ontology editor *TopBraid Composer*[11] has made the Manchester OWL Syntax the syntax of choice.

In summary, the Manchester OWL Syntax has been well received by non-logicians. However, it should be noted that while most users have found the syntax easy to read, in general, they still needed training to re-align their 'natural interpretation' with the correct OWL/DL interpretation. For example, it is often necessary to explain the precise meaning of 'some' – i.e. the semantics of existential restrictions. Another example is that universal restrictions are often interpreted to mean 'only and some' – trivial satisfaction of such restrictions is counter intuitive for many users. However, it is arguable that explanations of OWL semantics would be required whatever syntax was chosen. What is clear, is that the Manchester OWL Syntax has, to a large degree, ameleorated the 'prefix problem' (described in section 1.2) associated with restrictions. The syntax also seems to be more memorable than the German DL syntax, which means users get used to reading and writing class expressions in a shorter amount of time. They also find it easier to map between the syntax and semantics.

7 Conclusions

- The Manchester OWL Syntax is a new OWL Syntax that was designed in response to a demand from non-DL users for less logician like syntax.
- Key features of the syntax are that is uses natural language keywords rather than DL symbols, and an infix notation for restrictions.

[8] http://www.biopax.org

[9] The Brisith equivalent of the US Geological Survey – http://www.ordnancesurvey.co.uk/oswebsite/

[10] Personal communication with John Goodwin and O/S employees

[11] http://www.topbraidcomposer.com

– The syntax is suited for use in tools for presenting and editing class descriptions, for use in papers, e-mail messages and tutorials, where the audience is a non-DL audience. The syntax can also be used to represent complete OWL (including OWL-Full) ontologies.
– User feedback from groups such as BioPAX, Ontogenisis and the Ordnance Survey have confirmed that the Manchester Syntax is the preferred syntax for non-logicians when editing class expressions.
– A reference implementation of a Java based parser has been produced, which may be integrated into any tool. The implementation also includes a converter, which can transform the Manchester Syntax to RDF/XML syntax.

In summary, the OWL user community has indicated that it needs a range of syntaxes for a range of purposes. Each syntax discussed has its uses for a particular communtiy of either people or tools. In creating tools for non-logician users, the need for another syntax has been identified. The design of the Manchester OWL syntax has met the issues that have been discussed, and its wider user is recommend for testing and further development.

Acknowledgements

This work was supported in part by the CO-ODE project funded by the UK Joint In- formation Services Committee (JISC) and the HyOntUse Project (GR/S44686) funded by the UK Engineering and Physical Science Research Council. The authors would like to thank Andrew Gibson from the Bio Health Informatics Group at the University of Manchester for his input and feedback during the development of the Manchester OWL Syntax.

References

1. Knublauch, H., Musen, M.A., Rector, A.L.: Editing description logic ontologies with the Protégé-OWL plugin (2004)
2. Kalyanpur, A., Parsia, B., Hendler, J.: A tool for working with web ontologies (2005)
3. Becket, D.: Rdf/xml syntax specification (revised) (2004) http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.
4. Patel-Schneider, P.F., Hayes, P., Horrocks, I.: Owl web ontology language, semantics and abstract syntax. http://www.w3.org/TR/owl-semantics (2004)
5. W3C: The semantic web best practices and deployment working group. http://www.w3.org/2001/sw/BestPractices (2001)
6. Beckett, D.: New syntaxes for rdf. Technical report, Institute For Learning And Research Technology, Bristol (2004)
7. Rector, A., W.D., S., W.A., N., T.W., R.: A terminology server for medical language and medical information systems. Methods of Information In Medicine **34** (1994) 147–157
8. Sun: Javadoc. (http://java.sun.com/j2se/javadoc/)
9. 'Sreeni': Java compiler compiler [tm] (javacc [tm]) - the java parser generator. (https://javacc.dev.java.net)
10. Bechhofer, S., Volz, R., Lord, P.: Cooking the semantic web with the owl api (2003)