Sydney OWL Syntax - towards a Controlled Natural Language Syntax for OWL 1.1

Anne Cregan[1,2], Rolf Schwitter[3], and Thomas Meyer[1,2]

[1] NICTA, [Anne.Cregan,Thomas.Meyer]@nicta.com.au
[2] University of New South Wales, Australia
[3] Macquarie University, rolfs@ics.mq.edu.au

Abstract. This paper describes a proposed new syntax that can be used to write and read OWL ontologies in Controlled Natural Language (CNL): a well-defined subset of the English language. Following the lead of *Manchester OWL Syntax* in making OWL more accessible for non-logicians, and building on the previous success of Schwitter's PENG (*Processable English*), the proposed *Sydney OWL Syntax* enables two-way translation and generation of grammatically correct full English sentences to and from OWL 1.1 functional syntax. Used in conjunction with OWL tools, it is designed to facilitate ontology construction and editing by enabling authors to write an OWL ontology in a defined subset of English. It also improves readability and understanding of OWL statements or whole ontologies, by enabling them to be read as English sentences. It is hoped that by providing the option of an intuitive, easy to use English syntax which requires no specialized knowledge, the broader community will be far more likely to develop and benefit from Semantic Web applications. This paper is a discussion paper covering the scope, design, and examples of *Sydney OWL Syntax* in use, and the authors invite feedback on all aspects of the proposal via email to `krr.sydneysyntax@cse.unsw.edu.au`. Working drafts of the full specification are available at `http://www.ics.mq.edu.au/~rolfs/sos`.

1 Introduction

Following OWL reaching offical W3C recommendation status, a variety of notations for OWL class, property and individual descriptions and axioms became available through various tools, most notably Protégé [8] and SWOOP [6]. As noted in [3], these ranged from the officially recommended RDF/XML exchange syntax [2], through to a Description Logic style syntax, with Turtle/N-Triples [1] and the OWL Abstract Syntax [9] somewhere between the two extremes of verbosity and the specialized logical notation known as "squiggles" to non-logicians.

The experience of experts such as the Manchester Group in delivering OWL tutorials and workshops for domain experts identified that for the vast majority of non-logicians, none of the existing OWL syntaxes were suitable for writing class expressions and other types of axioms: they were either too verbose, or else the logical notation was intimidating and inconvenient to use. *Manchester OWL Syntax* [3] addressed these problems by providing an alternative syntax designed to be concise, without DL symbols, and quick and easy to read and write.

Manchester OWL Syntax has had substantial success and is reported to be the preferred syntax for non-logicians [3]. Discussion following its presentation at OWLED 2006 identified the potential, as a future goal for OWL, to extend the approach even further, to provide a syntax representing OWL in full English sentences. With a view to building on the previous success of Schwitter's *Processable English* (PENG) [11], which translates Controlled English to first-order logic, Cregan formed a small working group comprising the three Sydney-based authors (Cregan, Schwitter and Meyer) to design such a syntax.

The resulting proposed *Sydney OWL Syntax* is presented herein, and feedback is invited from all interested parties. As there are many design decisions to be made in such an undertaking, a large part of the paper is devoted to covering the design choices identified, and giving the rationale for the choices made. The syntax itself is presented via examples throughout the paper, but as space does not permit the inclusion of the emerging specification, readers should also consult the documentation available at `http://www.ics.mq.edu.au/~rolfs/sos`.

2 Background

2.1 Manchester OWL Syntax

Manchester OWL Syntax [3] is largely based on the German DL syntax and shares its compactness. Its key differentiating features are the replacement of special logical symbols such as $\exists$, $\forall$ and $\neg$ with the more intuitive keywords *some*, *only*, and *not*; the use of infix rather than prefix notation for keywords used in restrictions, preventing a misreading of class expressions found to be common amongst non-logicians; and the introduction of keywords such as **ValuePartition** facilitating common ontology design patterns. *Manchester OWL Syntax* has been reported to be well-received by non-logicians [3] and is the default syntax for Protégé-OWL and the commercially released OWL ontology editor *TopBraid Composer*[1]. In general, non-logicians have found it easier to grasp, remember and use than DL syntax. Although needing some training to re-align their natural interpretation of keywords to the correct OWL/DL interpretation, it successfully lowered the barrier for reading and interpreting ontologies.

Limitations: Although able to represent complete ontologies, *Manchester OWL Syntax* has been primarily designed for presenting and editing class expressions via tools, and representation / tool support for property and individual expressions seems to have had less focus. In addition, whilst certainly lowering the barrier, a syntax closer to English, with semantics matching a natural English interpretation could potentially remove it altogether.

2.2 PENG (Processable ENGlish)

PENG (Processable ENGlish) [11] is a machine-oriented controlled natural language (CNL) designed for writing unambiguous and precise specification texts for knowledge representation. Whilst easily understood by speakers of the base language, it has the same formal properties as an underlying formal logic language

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

and thus is machine-processable. It can be used, for example, for annotating web pages with machine-processable information [12]. PENG covers a strict subset of standard English, and is precisely defined by a controlled grammar and lexicon.

Specification texts written in PENG are incrementally parsed using a unification-based phrase structure grammar, and translated into first-order logic via discourse representation structures [7]. Standard first-order logic reasoning services are applied for reasoning tasks including consistency and informativity checking, and question answering.

As a brief example, the following sentences are written in PENG:

1. *If X is a research programmer then X is a programmer.*
2. *Bill Smith is a research programmer who works at the CLT.*
3. *Who is a programmer and works at the CLT?*

Sentence (*1*) describes a subclass relationship, sentence (*2*) asserts factual knowledge about a domain, and sentence (*3*) is used to query the terminological and factual knowledge expressed in (*1*) and (*2*). Standard first-order logic (FOL) query processing returns the answer *Bill Smith*.

The writing process of PENG is facilitated by predictive interface techniques: after the author enters a word form, the authoring tool displays look-ahead information indicating the available choices for the next word form, ensuring adherence to the lexicon and grammar. The author does not need to learn or remember the rules of the controlled natural language as these are taken care of by the authoring tool.

Limitations: The grammar of PENG is first-order equivalent and therefore more expressive than OWL 1.1. It is informed by FOL rather than DL considerations. In addition, the grammar has not been designed with bidirectionality in mind: PENG sentences are translated into FOL but not from FOL backwards into PENG. For these reasons, *Sydney OWL Syntax*, whilst informed by the learnings and experience of PENG, has essentially been designed from scratch. With regard to bidirectionality, Kaljurand and Fuchs [4] have presented a bidirectional mapping between a subset of OWL DL and Attempto Controlled English using a discourse representation structure as interlingua, but in recent work [5] they focus on one direction only: the verbalisation of OWL DL. Schwitter and Tilbrook [13] previously showed that there is no need for an interlingua and that bidirectionality can be achieved in a direct way using axiom schemas.

3 Scope

Sydney OWL Syntax has been scoped as follows:

1. **OWL 1.1 compatible**
   Unlike the 2004 OWL recommendation which uses a frame-like syntax convenient for manipulating ontologies by hand:

   ```
   ObjectProperty(hasAncestor domain(person) range(person))
   ```

   the emerging OWL 1.1 [10] has a functional-style syntax which breaks such axioms apart and makes them easier to manipulate programmatically:

```
ObjectPropertyDomain(hasAncestor person)
ObjectPropertyRange(hasAncestor person)
```

Sydney OWL Syntax takes OWL 1.1 functional syntax as the normative form for expressing OWL ontologies and the base form for translations. Combining readability and processability, it expresses the same information as:

```
If X has Y as an ancestor then X is a person.
If X has Y as an ancestor then Y is a person.
```

See Section 6 for considerations of conciseness in the design.

2. **Coverage of the entire OWL language**
   Anything that can be expressed in OWL 1.1 may be expressed in *Sydney OWL Syntax*. It provides complete coverage of all axioms and assertions that may be made in OWL 1.1, for example subproperty relations:

   ```
   If X has Y as a parent then X has Y as an ancestor.
   ```

   and property chains (= role composition):

   ```
   If X owns Y and Y has Z as a part then X owns Z.
   ```

3. **Two-way translation**
   Any OWL 1.1 ontology may be represented in *Sydney OWL Syntax* and conversely, ontologies constructed in *Sydney OWL Syntax* can be fully represented in any other OWL 1.1 syntax, without loss of information. The writing of ontologies in *Sydney OWL Syntax* is to be supported with interactive functionality such as look-ahead information, to assist the user and enforce syntactic validity.

4 Design goals

The key design goals of *Sydney OWL Syntax* are:

1. **Support non-logicians to build quality OWL ontologies**
   Support domain experts and analysts, particularly those without a logical background, to write good quality OWL ontologies. We assume that users are literate in English, and have at least an average ability to use a computer and think and express themselves logically in the normal sense of the word, but no specific knowledge of any formal notation is assumed.

2. **Provide English translations of OWL ontologies**
   Provide English translations of OWL ontologies which can be read and understood by English-speaking persons, without the need to refer to any other ontology syntax or representation. As with any ontology syntax, it is the responsibility of the author(s) to choose sensible and appropriate names for user-defined classes and properties.

3. **Modularity for future flavours of OWL**
   As OWL is an evolving language, and it is likely that new flavours of OWL corresponding to various formal logics will emerge, one of the design goals is a modular approach which facilitates contracting and expanding the syntax in correspondence with the logical operators to be included. For instance, words such as *must*, *may* and *cannot* are not currently used, as they correspond to

notions of permissibility and obligatoriness used by deontic logics. At some stage OWL may have a flavour based on a deontic logic, so these words are kept in reserve for that scenario.

4. **Implementable by OWL tools**
   Provide a specification which is sufficiently detailed and precise for implementation in ontology tools, as an alternative syntax to Manchester Syntax, OWL Abstract Syntax, and/or the other existing syntaxes. We note however that as OWL 1.1 functional syntax is not fully backwards-compatible with previous OWL syntaxes, the same applies for *Sydney OWL Syntax*.

5 Design choices

Whilst developing the syntax, several design decisions were encountered and choices made. We believe these decisions are ones which would be encountered by any effort to translate between a formal and a controlled natural language, and give the rationale for the choices made for *Sydney OWL Syntax*.

5.1 Naturalness versus closeness to OWL

A key decision was how natural we wanted the language to be. We observed a fundamental tradeoff between naturalness and closeness to OWL: on the one hand, the language could be more natural, but would lose its binding to OWL and thus become ambiguous and open to interpretation. This would seem to defeat the purpose of building an ontology as it is expressly for explicit logical representation of a domain. On the other hand, one can bind very tightly to OWL but this can result in some unnatural sounding English expressions, as there is often no exact or at least succinct equivalent in English for an OWL construction. For example,

```
hasFather is a FunctionalObjectProperty.
```

does not sound like a natural English expression, as firstly, it is an artefact of the ontology itself, and secondly, it uses abstract terms that are unknown to non-specialists. In contrast, *Sydney OWL Syntax* uses the terms of the application domain to convey the meaning without the need for any opaque encoding:

```
If X has Y as a father then Y is the only father of X.
```

In general we have opted towards tight binding with OWL 1.1 functional syntax whilst endeavouring to make the expressions as natural as possible.

1. **One or many CNL translations?**
   In natural language there are many ways to say the same thing - did we want to try to support all or a collection of them in translating a given OWL statement? For example, for the expression `SubClassOf(male, person)` should we support both *If X is a male then X is a person* and *Every male is a person* or allow one and only one CNL representation? We decided that for the first cut of *Sydney OWL Syntax*, there would be only one.
   Design choice: OWL syntax corresponds uniquely to *Sydney OWL Syntax*. That is, there is only one *Sydney OWL Syntax* form for each OWL form,

chosen to maximise succinctness and precision.

However, we appreciate the potential usefulness of supporting different modes of natural language expression for ontology construction purposes. For example, we have chosen to represent disjointness with the succinct `mutually exclusive`, but for clarification purposes it may be helpful to offer an expanded CNL translation such as *female or male is the case but not female and male.* One option is that such modes could be handled through the interface without formally being part of the syntax.
We also note that uniqueness of form refers to the syntactical form of the OWL statement, not its logical status - in some cases two OWL expressions may be logically equivalent but use different syntax, for example:
`DisjointUnion(person male female)` and

```
DisjointClasses(male female)
EquivalentClasses(person ObjectUnion(male female))
```

In this case, each syntactically distinct OWL expression corresponds to its own *Sydney OWL Syntax* equivalent:
`The class person is equivalent to male or female, and male and female are mutually exclusive.` and

```
The classes male and female are mutually exclusive. The
class person is fully defined as anything that is a male or
a female.
```

2. **How explicit should the OWL constructs be?**
One of the fundamental design decisions we faced was whether or not to talk about the ontology constructs themselves in the syntax. For instance, would a class axiom like `subClassOf(male, person)` translate to something like *There is a class called male which is a subset of a class called person* or to a statement about the domain itself, like *Every male is a person*? In the latter case, some OWL statements, such as `class male`, would have no translation at all in CNL as they as artefacts of the modelling process and don't assert anything about the domain itself.
Design choice: We opted to have limited explicit references to OWL constructs like classes and properties. As a consequence, some OWL axioms are not translated at all, but the knowledge is captured in *Sydney OWL Syntax* implicitly. Using a parsing process, any implicit concept in *Sydney OWL Syntax* can be unpacked into a corresponding OWL axiom. E.g., the translation of `Every male is a person` back to OWL produces a `class male` declaration and a subset axiom. Overall, all information from OWL is captured in *Sydney OWL Syntax* and given a *Sydney OWL Syntax* translation, the original OWL statements can be regenerated.

3. **Correspondence between OWL constructs and CNL constructs**
To facilitate modularity in respect of the addition or removal of logical op-

erators and constructions, we have carefully chosen grammar and lexicon to correspond tightly with the underlying logic, the aim being to implement as much modularity as possible within the boundaries of using natural grammar. For instance, the word *only* in

```
If X has Y as a son then Y is the son of only X.
```

is reserved for use in expressing functional or inverse functional properties, and not in any other context. By virtue of this tight binding, a person with familiarity with both OWL and *Sydney OWL Syntax* can read an ontology represented in *Sydney OWL Syntax* and recognise the OWL constructs via the words and phrases used.
Design choice: Where possible, each OWL construct has its own distinct natural language keyword or phrase.

4. **Use of linguistic and other background knowledge**
   Anaphoric reference: In natural language, it is common to refer to concepts introduced in previous statements via pronouns and definite noun phrases to refer to previously introduced entities. Note that to use the pronoun "he" requires previous knowledge of the referent being male. In an OWL context, such references require logical processing of other statements. In OWL ontologies statements are not necessarily in any order, so the entire ontology would need to be parsed.
   Number agreement: Linguistic background knowledge is also commonly used in natural language. For instance, the knowledge that the correct plural of *mouse* is *mice* is necessary to refer to *Three blind mice* instead of *Three blind mouses*. Adding an "s" to create plurals is a useful default rule but not always correct. However, we can use morphological rules and a list of exceptions as best approximation.
   Design choice: Each OWL statement is translated as a unit, without reference to any other statement in the ontology, or any other background or linguistic knowledge. Processing and using knowledge from outside the OWL statement vastly compounds the complexity of processing, thus has been avoided at the expense of providing anaphoric reference and safe number agreement.
5. **Use of variables**
   Whilst not a design preference, we found that some OWL statements could not be expressed clearly in CNL without using variables. If you need convincing, try as a test to express succinctly and unambiguously in English without using variables, the example involving role composition in section 3:

   ```
   If X owns Y and Y has Z as a part then X owns Z.
   ```

   One option we considered is to use phrases such as "something" and "something else" as pseudo-variables. But then one ends up with howlers such as the following: *If something owns something else and that something has another thing as a part then the original something owns that something.*
   Design choice: We decided to minimise the use of variables but found it impossible to do without them completely.

5.2 Complex constructs

Design choice: Complex class definitions are supported through an approach which supports nesting of expressions to any level. We plan to support expressions which use nesting up to three levels, for example:

```
The class old lady is partly defined as anything
  that has only cats as a pet
  and has some animal as a pet
  or has only gardeners as a lover.
```

5.3 Extra language support for user-defined terms

In building ontologies it is very common to use *has* and *is* combined with some other word or phrase when naming properties, e.g. *hasAge*; *isMotherOf* etc.
Design choice: *Sydney OWL Syntax* supports special processing of property names for *has* and *is* and their grammatical variants, providing camel case is used e.g. `isMotherOf` not `ismotherof`. This provides a more natural translation.

5.4 Definitions

Constructing correct definitions is challenging in OWL, since authors often fail to make a definition *complete* rather than *partial*. To address this problem we use the two markers `fully defined as` and `partly defined as` to indicate the logical status. For example, the following statement:

```
The class adult is fully defined as any person
  that has at least 20 as an age.
```

claims that the concept `adult` is fully defined by a set of necessary and sufficient conditions. The translation of this statement results in the subsequent functional-style syntax representation:

```
EquivalentClasses(adult
      ObjectIntersectionOf(Person DataAllValuesFrom(hasAge
      DatatypeRestriction(Datatype(xsd:nonNegativeInteger)
      owl:minInclusive "20"^^xsd:int))))
```

6 Design consequences

6.1 Tight binding to functional-style syntax

In general, the OWL 1.1 functional syntax requires more statements to express the same thing than the previous frame-like notation. The consequence of tight binding to the former is that *Sydney OWL Syntax* also has more statements. For instance, using the original OWL frame-like notation as a starting point, it would have been easier to translate the example given in Section 3 into one sentence rather than two:

```
If X has Y as an ancestor then X is a person and Y is a person.
```

6.2 Bidirectionality and context sensitive grammar

Sydney OWL Syntax is bidirectional, thus each statement translates into OWL functional-style syntax and vice versa, with the exception of statements of explicit OWL constructs, which have no *Sydney OWL Syntax* translation. An elegant way to achieve bidirectionality is to use a definite clause grammar and generate the output format during the parsing process [13]. In general, bidirectional translation requires a context-sensitive grammar. This may be illustrated as follows:

`If X has Y as a parent then Y has X as a child.` expresses an inverse relationship between two properties. Note that in the antecedent, the grammar needs to store the variable `X` in the subject position and the variable `Y` in the object position, whereas in the consequent their positions must be switched, otherwise we have a subproperty relationship. Additionally, the grammar has to provide a mechanism to absorb the auxiliary verb `has` and the prepositional objects `parent` and `child` into an OWL property name. To achieve bidirectionality, *Sydney OWL Syntax* will use a context-sensitive grammar which can store the required elements and employ an axiom schema which is instantiated during parsing: *InverseObjectProperties(Prefix1:Property1 Prefix2:Property2)*
For this example the schema looks as follows after parsing:
InverseObjectProperties(a:[has,parent],a:[has,child])
and this output can easily be transformed into the final format:
InverseObjectProperties(a:hasParent a:hasChild). In the ideal case the same grammar should accept this output and generate the original input sentence.

7 Conclusion and future work

Above we have set out the scope, design goals, decisions and choices informing the emerging *Sydney OWL Syntax* specification. The authors invite feedback on every aspect of the proposal, via email to `krr.sydneysyntax@cse.unsw.edu.au`. In parallel with collecting feedback from interested parties and moving towards a stable specification, we plan to start work on a demonstrator. In conclusion we note some features we envisage for tool interfaces.

The writing of an ontology in Sydney Syntax is to be supported by a predictive text editor which generates look-ahead information while a specification text is written [12, 14]. Thus the user does not need to learn the rules of the Sydney Syntax explicitly, since the writing process is guided by the text editor.

Such a text editor will be able to be used either in TBox mode, to express terminological axioms, or in ABox mode, to assert factual information about a specific domain. Once a set of terminological axioms has been specified, the resulting user-defined terminology can be used in ABox mode to specify instance data. From the terminological information available in the ontology, the text editor becomes “ontology-aware”, harvesting TBox input to generate new lookahead information guiding the writing process in ABox mode.

Acknowledgements

Research reported in this paper has been partially financed by the Macquarie University Centre for Language Technology (http://www.clt.mq.edu.au). We thank Phillip

Quinn and Matthew Horridge for their assistance in producing examples of OWL 1.1 functional syntax, and the members of the public-owl-dev@w3.org mailing list for their contributions. NICTA is funded by the Australia Government's Department of Communications, Information and Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Centre of Excellence program. It is supported by its members the Australian National University, University of NSW, ACT Government, NSW Government and affiliate partner University of Sydney.

References

1. D. Beckett. New syntaxes for rdf. Technical Report, 2004. Institute for Learning and Research Technology, Bristol.
2. D. Beckett. Rdf/xml syntax specification (revised). W3C Recommendation 10 February, 2004. At http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.
3. M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. H. Wang. The manchester owl syntax. In *Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006)*, 2006. available at http://owl-workshop.man.ac.uk/acceptedLong/.
4. K. Kaljurand and N. E. Fuchs. Bidirectional mapping between owl dl and attempto controlled english. In *LNCS 4187*, pages 179–189, 2006.
5. K. Kaljurand and N. E. Fuchs. Verbalizing OWL in Attempto Controlled English. In *Proceedings of OWLED07*, 2007.
6. A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and J. Hendler. Swoop: A 'web' ontology editing browser. *Journal of Web Semantics*, (4(2)), 2005.
7. H. Kamp and U. Reyle. *From Discourse to Logic.* Dordrecht: Kluwer, 1993.
8. N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng and O. Corby, editors, *Proc. of the 12th EKAW*, volume 1937 of *LNAI*, pages 17–32, Juan-les-Pins, France, 2000. Springer.
9. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. Owl web ontology language, semantics and abstract syntax, 2004. W3C Recommendation 10 February 2004, available at http://www.w3.org/TR/owl-semantics/.
10. P. F. Patel-Schneider and I. Horrocks. Owl 1.1 web ontology language overview, 2006.
11. R. Schwitter. English as a formal specification language. In *Proceedings of the Thirteenth International Workshop on Database and Expert Systems Applications (DEXA 2002)*, pages 228–232, 2002.
12. R. Schwitter and M. Tilbrook. Annotating websites with machine-processable information in controlled natural language. In *Advances in Ontologies 2006, Proceedings of the Second Australasian Ontology Workshop (AOW 2006)*, pages 75–84, 2006.
13. R. Schwitter and M. Tilbrook. Let's talk in description logic via controlled natural language. In *Proceedings of the Third International Workshop on Logic and Engineering of Natural Language Semantics (LENLS2006)*, pages 193–207, 2006.
14. C. W. Thompson, P. Pazandak, and H. R. Tennant. Talk to your semantic web. volume 9, pages 75–79, 2005.