A Controlled Natural Language Layer for the Semantic Web

Rolf Schwitter

Centre for Language Technology, Macquarie University,
Sydney, NSW 2109, Australia
schwitt@ics.mq.edu.au

Abstract. In this paper, I will show how a controlled natural language can be used to describe knowledge for the Semantic Web and discuss the formal properties of this language. At the first glance, the proposed controlled natural language looks like full English and can therefore be easily written and understood by non-specialists. However, its built-in grammatical and lexical restrictions, which are enforced by an intelligent authoring tool, guarantee that the language can be directly translated into description logic programs, i.e. the intersection of an expressive description logic with function-free logic programs. The controlled natural language can be used to make assertional and terminological statements as well as to specify rules for reasoning with the resulting assertional and terminological knowledge.

1 Introduction

The Semantic Web is based on the idea of having well-defined data on the Web linked in such a way that this data can be processed by computers to solve well-defined problems by performing well-defined operations. So far, a lot of effort has been put into the design of various standards by the W3C to extend the current Web to achieve this goal [13], but little attention has been paid to the so-called people axis where the Semantic Web is tailored for people, and not only for computers [10]. Even worse and somehow incomprehensible, it has been suggested, since the early days of the Semantic Web, that people should make an extra effort to understand computers, instead of asking computers to understand people's language [3].

It is true that full natural language is difficult to process for computers because of its inherent ambiguity and because of its expressive power which turns out to be one of the biggest stumbling blocks when it comes to machine processability. However, well-defined subsets of natural language (i.e. controlled natural languages) with clear grammatical and lexical restrictions can be used to describe a piece of knowledge in an unambiguous and precise way, if the authors are supported by an intelligent authoring tool that guarantees the compliance of the input with the controlled language standard [6,15].

Such a controlled natural language needs to fulfill at least three requirements to serve as a useful interface layer to the Semantic Web. Firstly, the controlled language should allow for expressing assertional and terminological statements as well as rules which can then be used for making inferences over the assertional and terminological knowledge. Secondly, the controlled language should be easy to write and read for humans without the need to memorise the grammatical and lexical restrictions of the language. Thirdly, the expressive power of the controlled language should be equivalent to a subset of first-order logic, in the ideal case a combination of description logic and clausal logic. In this paper, I will lay out the design of a controlled natural language that meets these requirements.

The reminder of this paper is organised as follows: In Section 2, I will give an overview of current Semantic Web ontology languages, describe their most important constructors and discuss research results that point out that the layering of these languages results in a number of problems. In Section 3, I will show that description logic programs offer a way out of these problems and provide a sound formal basis for language layering and for the design of a controlled natural language. In Section 4, I will introduce the controlled natural language by example and demonstrate how assertional, terminological and conditional statements can be expressed in a straightforward way in this language. In Section 5, I will discuss how the writing process of this controlled natural language is supported by an intelligent authoring tool. In Section 6, I will show how the controlled natural language is translated into the underlying representation language and then subsequently processed by a forward chaining model generator. In Section 7, I will talk about different reasoning tasks and their rendering in controlled natural language. Finally, in Section 8, I will conclude and summarise the advantages of the presented approach.

2 Semantic Web Ontology Languages

The most important ontology languages designed for the Semantic Web are currently RDF Schema [4] and the Web Ontology Language OWL [14]. The strategy of the W3C is to layer these ontology languages on top of RDF using XML as common syntax for the exchange and processing of metadata. RDF itself is based on the idea of identifying things on the Web via URI references and expressing statements about resources in terms of simple properties and property values, whereas each statement consists of a subject, a predicate, and an object. RDF models statements as nodes (for subjects and objects) and arcs (for predicates) in a graph. The vocabulary of such a graph is the set of names which occur as the subject, predicate or object of any triple in the graph [11].

2.1 RDFS

RDFS is a light-weight ontology language that was developed as the lowest layer of the Semantic Web languages. RDFS extends RDF to include larger vocabularies with more semantic constraints [4]. The most basic constructors of RDFS are used for the organisation of these vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classes. Apart from these basic constructors, full RDFS provides more powerful constructors that allow for meta-modelling and reification. It has been shown that these constructors lead to semantic problems, if the full power of RDFS is used to define the subsequent language layers and that the vocabulary needs to be strictly partitioned to be useful for these layers [2].

2.2 OWL

In comparison to RDFS, OWL adds more vocabulary for describing properties and classes: for example, specific relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes. OWL comes in three increasingly expressive layers that are designed for different groups of users: OWL Lite, OWL DL, and OWL Full [14]. If the RDFS vocabulary is strictly partitioned as required above, then a strict language inclusion relationship holds between RDFS, OWL Lite and OWL DL. However, no inclusion holds between OWL DL and OWL Full because of the lack of reification in OWL DL and OWL Lite. OWL Lite and OWL DL are syntactic variants of description logic languages, whereas OWL Full cannot be translated into a description logic language (since entailment in OWL Full is undecidable in the general case). Checking class membership and instance retrieval (ABox reasoning) is very expensive in most description logic reasoners and alternatives to description logic have been investigated. One way to solve this problem is to use a subset of description logic, translate it into function-free logic programs and rely on well-known deductive database techniques for reasoning [5].

3 Description Logic Programs

It has been argued that the intersection of description logic with logic programs can provide a straightforward computational pathway for reasoning and interoperability on the Semantic Web [7,8,9]. In particular, the description logic programming language $\mathcal{L}_0$ has been identified as the maximal intersection of the expressive description logic *SHOIN* (the language underlying OWL DL) and function-free logic programs (Datalog) [17]. This intersection does not allow arbitrary negation, disjunction in the head of rules or existential quantification. At first sight, this looks like a serious restriction, but it turns out that most of the ontologies developed for the Semantic Web fall under this fragment [17].

There exist restricted variants of OWL Lite and OWL DL (called OWL Lite$^-$ and OWL DL$^-$) that can be translated directly into Datalog. The RDFS subset of OWL Lite$^-$ corresponds to the RDFS subset of OWL Lite and provides the same basic constructors (as introduced above). Apart from that OWL Lite$^-$ allows for expressing complete class definitions, class and property equivalence, inverse and symmetric properties, transitive properties, and universal value restriction. OWL DL$^-$ adds very limited expressivity to OWL Lite$^-$ and allows for specifying classes based on the existence of particular property values and for restricting the first argument of the subclass property via existential value restriction or enumerated classes [5].

In contrast to the unrestricted versions of OWL Lite and OWL DL, the restricted variants stay within a well-established logic programming framework, allow the combination with rules languages and provide the basis for layering more expressive languages.

4 Towards a Controlled Natural Language Layer

Let us now work towards a controlled natural language that has the same formal properties as the description logic programming language $\mathcal{L}_0$. The user should be able to express statements and rules directly in controlled natural language using simple approved syntactic constructions that are familiar from full English. The semantic correspondence between the controlled natural language and the formal language is then guaranteed by a bi-directional translation.

Instead of writing the following assertional statement in OWL:

```
<owl:Thing rdf:ID="MarkHoyer"/>

<owl:Thing rdf:about="#MarkHoyer">
   <rdf:type rdf:resource="#Professor"/>
</owl:Thing>
```

the user should be able to express the same information in controlled natural language:

```
Mark Hoyer is a professor.
```

Here the subject 'Mark Hoyer' identifies the resource of the statement, the predicate 'is' identifies the property of the statement and the object 'professor' identifies the value of the statement. The namespaces for these names are not displayed here but are handled by the authoring tool that supports the writing process of the controlled natural language (see Section 5).

4.1 Assertional Statements in Controlled Natural Language

Assertional statements in controlled natural language will most likely be used by non-specialists who want to annotate a Web page with machine-processable information. The syntactic structure for making assertional statements consists of simple subject-predicate-object patterns and well-defined variations of these patterns, for example:

```
Ben Vermeer is a senior lecturer.
Ben Vermeer teaches COMP248 and COMP349.
Sharon Long who is trained by Mark Hoyer is a student.
```

The first statement is a simple one and consists of a single subject-predicate-object pattern. The second statement is a compound one and can (in principle) be split up into two simple statements by distributing the verb (i.e. the property):

```
Ben Vermeer teaches COMP248. Ben Vermeer teaches COMP349.
```

The third statement is a complex one that uses a relative sentence plus a passive construction. This statement corresponds to the following two simple statements:

```
Sharon Long is a student. Mark Hoyer trains Sharon Long.
```

As we will see in the next section, the relation between the active construction *trains* and the passive construction *is trained by* can be specified via a terminological statement that uses the *inverse of* constructor.

4.2 Terminological Statements in Controlled Natural Language

Terminological statements will most probably be used by knowledge engineers in order to construct an ontology. Terminological statements speak about classes, properties, instances of classes, and various kinds of relationships between instances, classes and properties. For example, the following statement

```
The property ‘teach’ has the type ‘object property’.
```

talks about the *‘teach’* property and assigns the type *‘object property’* to it. This defines the property *‘teach’* as a binary relation between instances of classes. The same terminological statement can be written in an abbreviated form:

```
‘teach’ has the type ‘object property’.
```

The property *‘teach’* can be further restricted by specifying its range and domain, for example:

```
‘teach’ has the domain ‘academic staff member’.
‘teach’ has the range ‘student’.
```

This allows us to relate instances of the class *‘academic staff member’* to instances of the class *‘student’*. More elaborate constructors are available that allow us to specify additional restrictions on classes and properties, for example:

```
‘teach’ has the equivalent property ‘give lessons to’.
‘train’ is a subproperty of ‘teach’.
‘train’ is the inverse of ‘be trained by’.
‘professor’ is a subclass of ‘academic staff member’.
‘senior lecturer’ is a subclass of ‘academic staff member’.
‘PhD student’ is a subclass of ‘student’.
‘COMP248’ has the equivalent class ‘Language Technology’.
```

This terminological knowledge is used by the model generator for reasoning purposes as well as by the authoring tool to guide the writing process of assertional statements.

4.3 Conditional Statements in Controlled Natural Language

As already mentioned, the description logic programming language $\mathcal{L}_0$ allows us to work with rules. These rules have the form of conditional sentences and can be used to specify class and property hierarchies, to describe property characteristics, and to indicate property restrictions. Constructing these rules is the task of a software engineer and not of a layperson.

Here is a rule in controlled natural language that specifies the formal meaning of the subclass relationship:

```
If C1 is a subclass of C2 and C2 is a subclass of C3
   then C1 is a subclass of C3.
```

This rule guarantees, for example, that if 'assistant professor' is a subclass of 'professor' and 'professor' is a subclass of 'academic staff member', then 'assistant professor' is a subclass of 'academic staff member'. Additionally, we need a rule that makes sure that an individual takes the class hierarchy into account:

```
If C1 is a subclass of C2 and E has the type C1
   then E has the type C2.
```

This rule ensures, for example, that if 'professor' is a subclass of 'academic staff member' and 'Mark Hoyer' has the type 'professor', then 'Mark Hoyer' has the type 'academic staff member'.

The following rule is an example of a property characteristic and specifies the formal meaning of the inverse property:

```
If E has the property P1 whose value is V
   and the property P2 is the inverse of P1
   then V has P2 whose value is E.
```

This rule says, for example, that if 'Mark Hoyer' has the property 'train' whose value is 'Sharon Long' and the property 'be trained by' is the inverse of 'train', then 'Sharon Long' has the property 'be trained by' whose value is 'Mark Hoyer'.

The following terminological statements define that the inverse property is a symmetric property and that the subject position and object position of the inverse property are realised by individuals:

```
'inverse of' has the type 'symmetric property'.
'inverse of' has the domain 'object property'.
'inverse of' has the range 'object property'.
```

Additional rules are required here that specify the behavior of the symmetric property and restrict its domain and range.

5 Writing in Controlled Natural Language

Writing in controlled natural language is supported by a intelligent text editor. This editor can be used either to write an assertional specification, to construct an ontology or to build an axiomatic rule set. The user does not need to learn the restrictions of the controlled natural language explicitly, since he is guided by a look-ahead mechanism while the text is written [16].

For an assertional specification, the user first selects the ontologies he wants to work with via a menu. Thereby the text editor becomes "ontology-aware", guides the writing process via look-ahead categories and deals with namespaces. Let's imagine that the user wants to express the assertional statement:

```
Ben Vermeer teaches COMP248.
```

The editor first displays a look-ahead category for the subject position:

[ProperNoun]

After entering the name 'Ben Vermeer', the editor displays further look-ahead categories (partially) derived from the syntactic information available in the grammar and from the terminological knowledge:

[who — is — has — org:teaches — edu:teaches — ...]

The user can now select the approved words from a context menu. In our example, the user has to choose between 'org:teaches' and 'edu:teaches', since there are two properties available with the same name defined in different namespaces. The property 'teaches' takes either instances of the class *'unit'* or of the class *'student'* as its range. After selecting the suitable option, the look-ahead editor asks for the value of the statement.

6 Processing Controlled Natural Language

Specifications written in controlled natural language are translated via a bi-directional Definite Clause Grammar (DCG) into a set of range-restricted clauses that can be processed directly by a forward chaining model generator [12]. These clauses (as well as facts) are uniformally represented in an implicational rule format. The following DCG rules

```
sentence(C1-C2) -->
  proper_noun(N,S),
  verb_phrase(a,_,N,S,C1-C2),
  ['.'].

verb_phrase(a,-,N,S,C1-[true--->term(S,V,O)|C2]) -->
  verb(N,V),
  noun_phrase(a,N,O,C1-C2).
```

illustrate how the rule (= fact)

```
true ---> term([‘Mark’,‘Hoyer’],rdf:[type],[professor]).
```

is built up via unification while the sentence

```
Mark Hoyer is a professor.
```

is parsed. This is a very simple example but the grammar can deal with more complex sentences. For instance, the following sentence

```
Ben Vermeer who is a senior lecturer teaches COMP248 and COMP349.
```

consisting of a relative sentence and a coordinated constituent is translated into three rules (= facts):

```
true ---> term([‘Ben’,‘Vermeer’],org:[teach],[‘COMP248’]).
true ---> term([‘Ben’,‘Vermeer’],org:[teach],[‘COMP349’]).
true ---> term([‘Ben’,‘Vermeer’],rdf:[type],[senior, lecturer]).
```

A terminological statement such as

```
‘teach’ has the domain ‘academic staff member’.
```

results in a similar translation:

```
true ---> term(org:[teach],rdfs:[domain],[academic,staff,member]).
```

And finally a conditional statement, such as

```
If P1 is a subproperty of P2 and P2 is a subproperty of P3
   then P1 is a subproperty of P3.
```

is translated into the following rule (clause):

```
term(P1,rdfs:[subPropertyOf],P2), term(P2,rdfs:[subPropertyOf],P3)
    ---> term(P1,rdfs:[subPropertyOf],P3).
```

The model generator takes such clauses as input and tries to generate a finite satisfying model (for details see [1]).

7 Reasoning in Controlled Natural Language

Given a number of assertional and terminological statements about the domain and a complete rule set that describes the meaning of the constructors, the model can be generated incrementally by performing forward chaining. Since the resulting model contains new information of the form:

```
term([Mark,Hoyer],rdf:[type],[academic,staff,member]).
```

It is straightforward to extract all entailed assertional statements and use the bi-directional DCG to render the output in controlled natural language, for example:

```
Mark Hoyer is an academic staff member.
Mark Hoyer gives lessons to Sharon Long.
Mark Hoyer teaches Sharon Long.
Mark Hoyer trains Sharon Long.
```

Not only can all entailed assertional statements be generated in this way, but also all terminological statements that are entailed, for example:

```
‘Language Technology’ has the type ‘unit’.
‘COMP248’ has the type ‘unit’.
‘Language Technology’ has the equivalent class ‘COMP248’.
‘give lessons to’ has the range ‘student’.
‘give lessons on’ has the range ‘unit’.
‘be trained by’ is the inverse of ‘train’.
```

The generated model can directly be used to answer questions in controlled natural language, for example:

```
Who trains Sharon Long?
Is Mark Hoyer an academic staff member?
Does Ben Vermeer teach Language Technology and COMP349?
```

Such questions are first translated into the implicational rule format (= negated clauses), then solution(s) are looked up in the model, and answers are generated in controlled natural language.

8 Conclusions

In this paper, I presented a controlled natural language for the Semantic Web that can be unambiguously translated into description logic programs, i.e. the intersection of an expressive description logic with function-free logic programs. In contrast to OWL, the controlled natural language cannot only be used for making assertional and terminological statements but also for describing axiomatic set of rules. Non-specialists can immediately use this language to annotate Web pages with machine-processable knowledge without the need to formally encode this information. Knowledge experts and software engineers can use this language to specify ontologies in domain-related terms and to define rules for reasoning. Note that the users of this controlled natural language do not need to remember the grammatical and lexical restrictions of the language, since these restrictions are enforced by an intelligent authoring tool that guides the writing process. In short: This controlled natural language does not require people to make an extra effort to understand computers as it is the case with formal languages designed for the Semantic Web, but asks computers to understand a well-defined subset of people’s language.

Acknowledgments

The research reported here is supported by the Australian Research Council, Discovery Project DP0449928. The author would also like to thank Marc Tilbrook for working on the look-ahead text editor.

References

1. S. Abdennadher, F. Bry, N. Eisinger, T. Geisler. 1995. The Theorem Prover Satchmo: Strategies, Heuristics, and Applications (System Description). *Research Report PMS-FB-1995-3*, May, Institute for Informatics, Ludwig Maximilians University, Munich, Germany.
2. G. Antoniou and F. van Harmelen. 2004. *A Semantic Web Primer.* MIT Press.
3. T. Berners-Lee. 1998. What the Semantic Web isn't but can represent. <http://www.w3.org/DesignIssues/>, September 17.
4. D. Brickley, R. V. Guha. 2004. RDF Vocabulary Description Language 1.0: RDF Schema. *W3C Recommendation*, 10 February 2004.
5. J. de Bruijn, A. Polleres, R. Lara, D. Fensel. 2004. WSML Deliverable, D20.1 v0.2, OWL$^-$. *WSML Working Draft*, May 25.
6. N. E. Fuchs, U. Schwertel, R. Schwitter. 1999. Attempto Controlled English - Not Just Another Logic Specification Language. Vol. 1559 of *LNCS*, Springer.
7. B. N. Grosof, I. Horrocks, R. Volz, S. Decker. 2003. Description logic programs: Combining logic programs with description logic. In: *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pp. 48–57.
8. I. Horrocks, P. F. Patel-Schneider. 2003. Three theses of representation in the semantic web. In: *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, pp. 39–47.
9. I. Horrocks, P. F. Patel-Schneider. 2004. A proposal for an OWL rules language. In: *Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004)*, pp. 723-731.
10. M. Marchiori. 2004. Towards a People's Web: Metalog. In: *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004)*, IEEE Press, pp. 320-326.
11. F. Manola and E. Miller. 2004. RDF Primer. *W3C Recommendation*, 10 February 2004.
12. R. Manthey and F. Bry. 1988. Satchmo: a theorem prover implemented in prolog. In: E. Lusk and R. Overbeek, (eds.), *Proceedings of CADE-88*, vol. 310 of LNCS, Springer, 415–434.
13. E. Miller, R. Swick, D. Brickley, B. McBride, J. Handler, G. Schreiber, D. Connolly. 2005. Semantic Web. *W3C, Technology and Society, Semantic Web Activity*, 19 August 2005.
14. M. K. Smith, C. Welty, D. L. Mc Guinness. 2004. OWL Web Ontology Language. Guide. *W3C Recommendation*, 10 February 2004.
15. R. Schwitter, A. Ljungberg, D. Hood. 2003. ECOLE – A Look-ahead Editor for a Controlled Language. In: *Controlled Translation, Proceedings of EAMT-CLAW03*, May 15-17, Dublin City University, Ireland, pp. 141–150.
16. R. Schwitter and M. Tilbrook. 2004. Controlled Natural Language meets the Semantic Web. In: A. Asudeh, C. Paris, S. Wan (eds.), *Proceedings of the Australasian Language Technology Workshop 2004*, Macquarie University, pp. 55-62.
17. R. Volz. 2004. Web Ontology Reasoning with Logic Databases. *PhD thesis*, AIFB, Karlsruhe.