Problems with OWL Syntax

Boris Motik and Ian Horrocks

University of Manchester
Manchester, UK

Abstract. In this paper we discuss three problems with OWL syntax that repeatedly surface in practice. The first problem is that OWL does not allow for explicit *declarations*—assertions that a certain class, property, or an individual exists in an ontology. This aspect of the OWL standard was often misinterpreted, which caused design errors in OWL APIs; moreover, the lack of declarations makes devising an intuitive structural consistency check for OWL ontologies difficult. The second problem is that OWL Abstract Syntax and OWL RDF syntax rely on the separation between object and data property names for disambiguation. We show that this prevents an unambiguous interpretation of certain syntactically well-formed OWL ontologies; furthermore, it makes implementing OWL parsers unnecessarily difficult. The third problem is that OWL Abstract Syntax cannot be translated into OWL RDF syntax without loss of information. We present possible solutions to these three problems, which, if adopted in OWL 1.1, would lead to a cleaner standard and would significantly simplify the implementation of OWL APIs.

1 Introduction

The Web Ontology Language (OWL) has successfully been applied to many practical problems in computer science in the past couple of years. However, discussions in the developer community and our own experience in building ontology management systems for OWL have revealed several problems with all three OWL syntaxes: OWL Abstract Syntax [5], OWL XML [4], and OWL RDF [2]. These issues were often misinterpreted by the developers, which caused design errors in OWL systems. The users of OWL APIs often try to find ad hoc solutions to these problems, which easily leads to further errors and performance penalty in practical applications. Similar issues were discussed for DAML+OIL—a precursor of OWL—in [1].

The first problem is that OWL does not provide for explicit *declarations* of ontology elements. For example, it is not possible to explicitly asset in an OWL ontology that a class exists. This may come as a surprise, since certain constructs of OWL syntax have a flavor of declarations. However, as we discuss in Section 2, a strict analysis of the OWL standard reveals that these constructs actually do not provide a satisfactory mechanism for declaring ontology entities. Apart from causing confusion with designers of OWL APIs, the absence of declarations makes it difficult to devise an intuitive check for structural consistency of OWL ontologies (e.g., to check whether a class has been declared before it is used).

The second problem is that, as we discuss in Section 3, OWL syntax is not completely context-free. The syntactic form of axioms for object and data properties is identical, and, to disambiguate the two, OWL relies on the separation between the names of object and data properties. This makes the OWL syntax ambiguous: as we already mentioned, OWL does not require all names to be declared, so it may happen that an ontology does not contain sufficient information to correctly disambiguate the axioms. Moreover, this makes implementing parsers for OWL quite difficult since parsing essentially requires two passes: one to "guess" the types of names used as properties and the other to actually generate the ontology axioms.

The third problem is that OWL Abstract Syntax is structurally different from the OWL RDF syntax—an ontology written in the former syntax cannot be translated into an ontology written in the latter syntax without loss of information about the structure of axioms. This exacerbates the two problems mentioned in previous paragraphs.

In this paper we present possible solutions to these problems. A particular challenge is to achieve backwards compatibility with the existing version of OWL. We believe that incorporating (a possibly modified variant of) these solutions in OWL 1.1—a recently proposed extension of the OWL standard—would make the standard clearer and would simplify the implementation of practical systems.

2 The Lack of Declarations

It OWL, it is not possible to *declare* a class—that is, to state its existence in an ontology. This may surprise the reader: OWL Abstract Syntax allows for the statement

(1) Class(Person partial Animal)

which intuitively looks like a declaration of the Animal class; that is, most OWL users will interpret this as a statement that the class Person exists in the ontology and that it is a subclass of the class Animal.

However, statements of such form do not really fulfill the usual intuitive requirements on declarations known from, say, programming languages. The same ontology can legally contain the following statement as well:

(2) Class(Person partial Homosapiens)

This ontology now seemingly contains two declarations for Animal, which is counterintuitive. OWL syntax contains many redundancies: the second axiom can be equivalently stated as follows:

(3) SubClassOf(Person Homosapiens)

This form (3) corresponds to the encoding of (2) in description logics (DLs)—the formalism underlying OWL providing for its semantics. The form (2) was introduced to provide a frame-like syntactic flavor for OWL. Namely, frame-based

knowledge representation formalisms, such as Open Knowledge Base Connectivity (OKBC) [3], typically allow the users to collect all relevant information about a class into a frame, which is uniquely associated with a class. The syntax of OWL resembles the syntax of frame-based languages; however, OWL as a langauge is fundamentally different from such languages. The statements (1) and (2) are *class axioms* and not frame definitions. Axioms merely specify constraints on the first-order interpretations of the ontology, and they do not have any extralogical meaning, such as asserting existence of an entity.

An explicit declaration mechanism would be quite useful in OWL. The main usage of such a mechanism is to provide a basis for a structural ontology consistency check. For example, the user might inadvertently type Animla instead of Animal; since OWL does not require classes to be explicitly declared before they are used, it is not possible to detect this error in a consistent way. To remedy this, some OWL management APIs interpret axioms such as (1) as declarations; however, by doing so, they necessarily deviate from the standard.

We now discuss three possible solutions to this problem, together with their pros and cons.

2.1 Solution 1: No Declarations in OWL

The first possibility is to leave OWL as it is—that is, not to introduce explicit declarations of entities at all. In this case, the standard should explicitly state that statements of the form (1) and (2) are axioms and should not be misinterpreted as declarations, that an ontology is allowed to contain several axioms for the same class, and that there is no way to structurally check the consistency of an ontology.

We point out the implications that such a decision would have on the design of the APIs for management of OWL ontologies. In particular, an API that strictly follows the structure of the OWL standard should not try to mimic the frame-like systems; rather, it should distinguish the axioms (1) and (2) from the following semantically equivalent but syntactically distinct axiom:

(4) Class(Person partial Animal Homosapiens)

Furthermore, such an API should *not* provide operations such as "create a class and add it to the ontology." Consider the case where a user creates a class C, adds it to the ontology, saves the ontology into a file, and then loads the ontology back. Intuitively, the ontology should still contain C; however, this is difficult to ensure because C does not occur in any axiom and the OWL syntax cannot represent class declarations. Similarly, an OWL editor based on such an API could *not* provide an operation for creating ontology entities.

2.2 Solution 2: Interpret Class Axioms as Declarations

The second possibility is to interpret class axioms as declarations. In this case, the OWL standard should be modified to enforce the structural consistency of declarations and usages of entities:

– An ontology O should be allowed to contains at most one declaration for an entity α.
– If an ontology O contains a declaration for an entity α, then no ontology O' different from O and imported by O (either directly or indirectly) should contain a declaration for α.
– If an entity α is used in an axiom in an ontology O, then α should be declared either in O or in an ontology O' imported by O (either directly or indirectly).

An API for management of OWL ontologies could now provide operations such as creating a class without any problems. For example, creating a class Person would result in adding the following declaration to the ontology:

(5) Class(Person partial)

Furthermore, the API might provide the operations for modifying this declaration, such as "add a superclass." For example, adding Animal as a superclass to Person would result in replacing (5) with the following declaration:

(6) Class(Person partial Animal)

Such a solution is much closer to the frame-based systems. Whereas it is relatively clean and has minimal impact on the existing standard, it would introduce backwards incompatibility, since many existing OWL ontologies would become structurally invalid.

2.3 Solution 3: Introduce Explicit Declaration Axioms

The drawback regarding backward compatibility mentioned in the previous section can be corrected by introducing a novel type of "declaration" axioms. In such a case, the axioms such as (1) and (2) would still be allowed to occur in one ontology; however, OWL standard would be extended with explicit statements stating that an entity exists in an ontology. For example, one might state the following declaration axiom:

(7) DeclareClass(Animal)

If such a solution were adopted, class axioms of the form (1) would behave as explained in Section 2.1, whereas existence axioms would behave as explained in Section 2.2.

3 Typing Object and Data Properties

The syntax of OWL is not context-free: the meaning of some axioms is not defined solely by their syntactic form. Consider the following axiom:

(8) SubPropertyOf(A B)

The syntactic form of (8) does not tell us whether A and B should be treated as object or as data properties. The names for object and data properties are required to be distinct in the OWL DL and OWL Lite variants of the language, which is used to disambiguate axioms such as (8). The OWL Full variant does not distinguish data from object properties, which makes implementing OWL Full quite difficult in practice.

Such a definition of OWL exhibits a conceptual problem that is strongly related to the issues presented in Section 2. Namely, OWL does not require A and B to be declared before they are used. Hence, the ontology containing only the axiom (8) should not be considered to be correct, since it does not contain enough information to disambiguate the axiom (8). However, this contradicts the prevailing idea of OWL that declarations are not necessary. Therefore, we consider this to be an incompleteness in the current OWL standard.

Current APIs for management of OWL ontologies solve this problem by looking for other occurrences of properties A and B that might help disambiguate the syntax. For example, if the ontology additionally contains the axiom (9), then the restriction on A suggests that A is a data property; combined with (8) one can also assume that B is a data property as well.

(9) Class(C partial restriction(hasValue(A 1.0)))

Clearly, such an approach is unnecessarily complex. Parsing an ontology requires two passes: one to determine the types of objects and one to actually create the ontology axioms. This is further complicated if an ontology imports another ontology. Namely, in some (quite realistic) cases, to disambiguate a name, one should not only look at the text of the ontology being parsed, but also at the ontologies that are imported; since imports are allowed to be cyclic, the overall problem suddenly becomes quite complex. Such a solution necessarily results in performance drawbacks that are significant in cases of large ontologies.

Many developers of applications have repeatedly pleaded for the possibility of using the same name to denote classes, properties, and individuals. Due to popular demand, this will be allowed in OWL 1.1 by a feature known as *punning*. However, punning makes disambiguating axioms even more complex, because the same name may be used for object and data properties simultaneously.

Similar ambiguities arise not only in property inclusion axioms but also in property restrictions. For example, from the following axiom it is not clear whether the restriction is on object or on data property:

(10) Class(C partial restriction(someValuesFrom(A D)))

To disambiguate (10), one should first determine that D is a class, which then allows one to conclude that A is an object property.

Apart from OWL Abstract Syntax, OWL RDF syntax also suffers from this problem. For example, the axiom (8) is represented in OWL RDF as the following RDF triple:

(11) $\langle$A, rdfs:subPropertyOf, B$\rangle$

By looking only at (11), one cannot tell whether A and B should be considered to be object of data properties. On the contrary, OWL XML syntax is fully context-free. For example, if the axiom (10) talks about object properties, if would be written in OWL XML as follows:

```
<SubPropertyOf sub="A">
  <ObjectProperty name="B" />
</SubPropertyOf>
```

We strongly believe that the ambiguities mentioned in this section should not be present in a widely accepted industrial standard. Furthermore, a lot would be gained by modifying the standard in a way that would allow for simpler implementations. Therefore, we propose the OWL syntaxes to be changed in OWL 1.1 as follows.

3.1 Amendments to OWL Abstract Syntax

We propose the following changes to be made to OWL Abstract Syntax:

- The SubPropertyOf grammar terminal should be replaced with SubDataPropertyOf and SubObjectPropertyOf terminals.
- The restriction grammar terminal should be replaced with dataRestriction and objectRestriction terminals.

For example, if the axiom (8) should talk about object properties, it would be written in the new syntax as follows:

(12) SubObjectPropertyOf(A B)

3.2 Amendments to OWL RDF Syntax

We propose the following changes to be made to OWL RDF syntax:

- New RDF properties owl:subDataPropertyOf and owl:subObjectPropertyOf should be introduced and used to specify inclusion of object and data properties, respectively.
- New RDF properties owl:dataRestriction and owl:objectRestriction should be introduced and used to specify data and object restrictions, respectively.

To achieve backward compatibility at the level of the RDF semantics, one might make owl:subDataPropertyOf and owl:subObjectPropertyOf subproperties of the property rdf:subPropertyOf; similarly, one might make owl:dataRestriction and owl:objectRestriction subproperties of owl:restriction. This would preserve compatibility with OWL Full semantics, but would allow us to define the semantics for OWL DL and OWL Lite in a clean way.

3.3 Backward Compatibility Issues

Clearly, the proposed changes are backwards incompatible with the existing versions of OWL standards. We do not consider this to be a significant issue for the OWL Abstract Syntax: this syntax was not intended to be implemented in practical systems, so it has not been used widely, and we believe that it can be changed without a big impact on the user community.

The vast majority of existing OWL ontologies has been created in OWL RDF syntax; therefore, changes to this syntax are more likely to cause problems. Therefore, we propose the following migration path for OWL RDF: the existing constructs should still be allowed, but should be marked as deprecated. In this way, users might gradually update their ontologies to the new syntax, and the deprecated syntax elements might be completely removed in the next revision of the OWL language.

3.4 OWL RDF Syntax and Annotations

OWL RDF syntax suffers from one further problem, which does not occur in OWL Abstract Syntax. Consider the triple (13):

(13) $\langle \mathsf{S}, \mathsf{P}, \mathsf{O} \rangle$

If O is a RDF literal, we may conclude that P is a data property; similarly, if P is a RDF resource, we may conclude that P is an object property. However, an ambiguity arises in OWL RDF because P might be an annotation property.

We do not have an elegant solution to this problem. Triples in which P is a built-in annotation property, such as rdfs:label, can be disambiguated by looking at the property name. For user-defined annotation properties, a reasonable solution is to say that all annotation properties must be explicitly declared at the beginning of the the ontology where they are used using an explicit triple of the following form:

(14) $\langle \mathsf{P}, \mathsf{rdf{:}type}, \mathsf{owl{:}AnnotationProperty} \rangle$

4 Mismatches Between OWL RDF and Abstract Syntax

OWL RDF and the Abstract Syntax do not match in structure. Namely, the RDF triple

(15) $\langle \mathsf{A}, \mathsf{rdfs{:}subClassOf}, \mathsf{B} \rangle$

corresponds in OWL Abstract Syntax to one of the following two axioms:

(16) $\mathsf{SubClassOf(A\ B)}$

(17) $\mathsf{Class(A\ partial\ B)}$

Clearly, saving an ontology written in OWL Abstract Syntax into a file using OWL RDF syntax results in a loss of the information about the structure of the ontology axioms.

It is tempting to say that, assuming that the RDF triple (15) occurs alone in an ontology, then (15) corresponds to the axiom (16); for (15) to correspond to (17), the ontology should additionally contain the triple (18):

(18) $\langle \mathsf{A}, \mathsf{rdf{:}type}, \mathsf{owl{:}Class} \rangle$

However, such a solution is not possible if the OWL standard is to be interpreted precisely. Namely, the triple (18) is implied as soon as the class name A occurs in a class restriction.

This problem further exacerbates the problem of missing declarations outlined in Section 2. If the solution from Section 2.2 were adopted (i.e., if the axioms of the form (17) are treated as declarations of OWL entities) in OWL Abstract Syntax, we still cannot unambiguously express declarations in OWL RDF syntax.

To remedy this situation, we propose to extend the OWL standard with a section providing an explicit translation between the two syntaxes. We have not yet fleshed out the details of such a translation; however, in the rest of this section we outline what such a translation might look like.

A translation would be defined to consider only explicitly present triples, and not the implied triples. This would allows us to use triples of the form (18) as declarations: if a triple of the form (18) exists for a class A, the translation produces an axiom of the form (18).

Full correspondence between the two syntaxes is unlikely to be possible. Consider the following set of triples:

(19) $\langle \mathsf{A}, \mathsf{rdf{:}type}, \mathsf{owl{:}Class} \rangle$

(20) $\langle \mathsf{A}, \mathsf{rdfs{:}subClassOf}, \mathsf{B} \rangle$

(21) $\langle \mathsf{A}, \mathsf{rdfs{:}subClassOf}, \mathsf{C} \rangle$

Even if we interpret (19) as class declaration, we can interpret (20) and (21) as (22), (23), or (24):

(22) $\mathsf{SubClassOf}(\mathsf{A\ B})$ and $\mathsf{Class}(\mathsf{A\ partial\ C})$

(23) $\mathsf{Class}(\mathsf{A\ B})$ and $\mathsf{SubClassOf}(\mathsf{A\ partial\ C})$

(24) $\mathsf{Class}(\mathsf{A\ partial\ B\ C})$

To make the translation unambiguous, we propose to design a set of heuristics. In the presented example, the form (24) is likely to be a satisfactory solution.

5 Conclusion

In this paper we identify three drawbacks in the current definition of the OWL syntax. The first drawback is that OWL does not provide for explicit declarations

of ontology entities. This caused numerous misunderstandings in the past, and it makes defining an intuitive structural integrity check for OWL ontologies very difficult. The second drawback is related to the fact that OWL Abstract Syntax and OWL RDF syntax are not context-free. Namely, OWL currently relies on a separation between the names of object and data properties to disambiguate the syntax. Combined with the lack of explicit declarations of ontology entities, this problem actually makes the OWL specification incomplete and difficult to implement in practical systems. The third problem is caused by the fact that OWL Abstract Syntax and OWL RDF syntax do not correspond in structure.

We have discussed possible solution to each problem mentioned. Based on the feedback from the community of OWL users and developers, we hope that some of these solutions will find their way into the OWL 1.1 revision of the OWL standard.

References

1. S. Bechhofer, C. A. Goble, and I. Horrocks. DAML+OIL is not Enough. pages 151–159, Stanford University, CA, USA, July 30–August 1 2001.
2. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference, W3C Recommendation, February 10 2004.
   http://www.w3.org/TR/owl-ref/.
3. V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice. Open Knowledge Base Connectivity 2.0.3, April 9 1998.
   http://www.ai.sri.com/ okbc/okbc-2-0-3.pdf.
4. M. Hori, J. Euzenat, and P. F. Patel-Schneider. OWL Web Ontology Language: XML Presentation Syntax, W3C Note, June 11 2003.
   http://www.w3.org/TR/owl-xmlsyntax/.
5. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language: Semantics and Abstract Syntax, W3C Recommendation, February 10 2004.
   http://www.w3.org/TR/owl-semantics/.