The Consistency of OWL Full

Jeremy J. Carroll[1] and Dave Turner[2]

[1] HP Labs, Bristol, UK
[2] Computer Laboratory, Cambridge University
Dave.Turner@cl.cam.ac.uk

Abstract. We show that OWL1 Full without the comprehension principles is consistent, and does not break most RDF graphs that do not use the OWL vocabulary. We discuss the role of the comprehension principles in OWL semantics, and how to maintain the relationship between OWL Full and OWL DL by reinterpreting the comprehension principles as permitted steps when checking an entailment, rather than as model theoretic principles constraining the universe of interpretation. Starting with such a graph we build a Herbrand model, using, amongst other things, an RDFS ruleset, and syntactic analogs of the semantic "if and only if" conditions on the RDFS and OWL vocabulary. The ordering of these steps is carefully chosen, along with some initialization data, to break the cyclic dependencies between the various conditions. The normal Herbrand interpretation of this graph as its own model then suffices. The main result follows by using an empty graph in this construction. We discuss the relevance of our results, both to OWL2, and more generally to a future revision of the Semantic Web recommendations.

1 Introduction

1.1 A Wart on the Face of the Semantic Web

The lack of a consistency proof for OWL Full is a wart on the face of the, otherwise impressive, formal foundations of the Semantic Web. This paper does not turn that wart into a beauty spot. Instead, we propose a line of cosmetic surgery, and provide the first necessary pre-op preparation.

We first address the reader who is already thinking of giving up, because semantics is hard, OWL Full is hard and OWL Full semantics is very hard. Before moving on to the next paper, we hope that the conclusions to this paper will partially address your concerns, and urge you to read this subsection and then skip straight to the conclusions.

This paper is motivated by two things. First, the authors and presumably others have tried, and failed over an extended period of time to prove that OWL Full is consistent. This is a minimal requirement for the OWL Full semantics to have any meaning at all. For example, if OWL Full is not consistent, then theorem 2 of [1], which relates OWL DL to RDF via OWL Full, is trivially true, and makes no claim. Our view is that the Semantic Web involves both RDF and description logics, and the place that these come together is in OWL Full. Thus,

we believe OWL Full is important. But also we have found that OWL Full is simply too difficult. No one implements OWL Full, not even close. It is not simply that OWL Full is formally undecidable, so a logically complete implementation is impossible but much more significantly that many of the apparently motivated aspects of OWL Full have no relationship with any implementation experience (including the implementation experience of the description logic community).

Second, Michael Schneider recently found a new variant of the Patel-Schneider paradox. This shows that the comprehension principles (see section 2) used in OWL1 Full, cannot be extended in a natural way to cover OWL2 [2].

We prove that OWL Full without the comprehension principles is consistent and doesn't break RDF or RDFS (which we define in a formal way). We sketch how theorem 2 of [1], which currently relies on the comprehension principles, can be rescued, by recasting them in a syntactic rather than semantic form. Should OWL Full be proved inconsistent, this gives a plausible path to recover it. Since implementations do not implement the comprehension principles, this would have no impact on users or implementers of OWL.

Our proof is instructive, in that it provides the first RDFS interpretation that satisfies many of the conditions of the OWL Full semantics. By understanding this paper and the separate appendix [3] which presents the approximately 15,000 triples that articulate that interpretation, we can get a much better feel for what the OWL Full semantics actually says.

Even without the comprehension principles, OWL Full is still much too complicated. In the closing sections we propose further surgery.

1.2 Notation

We use dom and ran as the domain and range of functions (considered as sets of pairs). id_X is the identity function on X. To accommodate line length constraints we shorten some of the RDF, RDFS and OWL vocabulary. In particular, we omit all namespace prefixes, except for `rdfs:Class` vs `owl:Class`. We also shorten some of the names in the namespaces, e.g.: `Prop`*erty*, `subProp`*erty*`Of`, `equiv`*alent*`Class` and `priorVers`*ion*.

1.3 Outline of Paper, and Additional Material

The paper starts by discussing the rationale behind the comprehension principles, and their rôle. In section 3 we define OWL Full_c, which does not contain them. The next several sections build up to section 9, in which we build a Herbrand graph. This follows the method of [4, 5] of building a graph in which the property extension of every predicate is made completely explicit. For example, since `Nothing` and `Thing` are different in OWL there is a triple (`Thing`, `differentFrom`, `Nothing`) in the Herbrand graph. A semantic interpretation of the graph in terms of itself, is given in section 10. In section 11, we sketch how the functions of the comprehension principles can be replaced, and conclude with a proposal for moving forward with OWL.

This is a short version of a longer technical report [6]. Proofs are nearly always omitted here as are several definitions, lemmas and more mathematical discussions. Theorem, definition and lemma numbering is the same in the two versions, resulting in some gaps in the number sequence in this paper.

2 The Comprehension Principles

Consider:

$$A = \{1\} \qquad B = \{2\} \qquad C = \{1, 2\} \tag{1}$$

Given (1), then, in classical logic:

$$C = A \cup B \tag{2}$$

Both (1) and (2), can be expressed in a natural way in OWL. In the direct semantics of OWL DL, and the semantics of OWL Full, we have the entailment analagous to:

$$(1) \Rightarrow (2) \tag{3}$$

However, the OWL Full entailment depends on the comprehension principles, so that OWL Full$_c$ which we define in section 3, does not provide this entailment. The problem is that the $A \cup B$ is written in OWL with a list (an RDF Collection), and both of the list cells correspond to resources in the domain of discourse (or universe), as specified in the RDF Semantics. In OWL Full$_c$, as in RDF and RDFS, no requirement for these lists to exist is made, so that in general, they might not, and the entailment fails.

Patel-Schneider [7] considered such problems at the beginning of the development of OWL. Discussing restrictions rather than lists, he wrote: "OWL interpretations must include resources for many restrictions, essentially all the restrictions that can be built". Partly because he took the lead rôle in designing the OWL semantics and partly because of the strength of his argument this view became embedded in the OWL Full semantics as the comprehension principles which require OWL Full interpretations to include "essentially all" restrictions and lists. An OWL Full interpretation of even the empty graph, containing no triples at all, is immensely complicated. For example, there is a `someValuesFrom` restriction for every class and every property, and since each of these is itself a class (as are all the other restrictions, complements, unions and intersections that are required by other comprehension principles), then there is an infinite productive power, prepopulating the universe with all such objects recursively.

2.1 The Patel-Schneider and Schneider paradoxes

That paper [7] also reports the Patel-Schneider paradox, which showed that if "essentially all" was made too general, then logical disaster loomed. He constructed a self-referential restriction that "consists of all resources that do not belong to it". If this were part of the prepopulated universe, then this would be

a paradox following the form of Russell's paradox. The comprehension principles of OWL carefully avoid that particular problem, by not requiring self-referential comprehension. Thus the restriction from the Patel-Schneider paradox, is simply a falsehood, rather than a paradox, under the OWL Full semantics. However, without a consistency proof, there is no guarantee that other logical paradoxes are not implicit within the complex tangle of resources required by the comprehension principles.

Recently, Michael Schneider [8] found a new variant of the Patel-Schneider paradox, for OWL2. The new self-restriction construct of OWL2 [2] is the class on which a given property is reflexive. By its very nature the self-restriction on `type` is self-referential and its complement is a new formulation of Patel-Schneider's paradoxical class.

2.2 Interdependencies

The comprehension principles lead to a tangle of interdependencies between the infinitely many resources they require because some of them concern properties and classes used within the framework of OWL. Restrictions on the predicates `type` or `onProperty` are particularly problematic, or on many of the other predicates in logical vocabulary. The normal way of trying to show that a model theory is consistent is to actually produce an interpretation that satisfies the constraints, for example by modifying an interpretation from some other model theory already known to be consistent. Cyclic interdependencies makes this harder as we will see in the construction used in this paper. The cyclic interdependences amongst the comprehension principles are sufficiently severe to make it very difficult to break out of them enough to construct an interpretation.

Given these difficulties, we decided to tackle the easier problem of OWL Full without the comprehension principles. This leaves us the task of providing some account of the non-entailment (3), see section 11.

3 OWL Full$_c$

OWL Full$_c$ is OWL Full without the comprehension principles. An OWL Full$_c$ interpretation is defined as a D-interpretation that satisfies all the constraints of an OWL Full interpretation except for the comprehension principles. There are two minor omissions from [1], that we provide in [6].

When referring to [1], it should be remembered that in OWL Full, we have the simplifications: IOT $= R_I$, IOOP $= P_I$ and IOC $= C_I$.

4 Generalized Graphs

We follow [9] and simplify the concept of RDF graph [10]:

Definition 1. *A* generalised graph *is any subset of* $(L \cup U \cup B)^3$*, with* L, U, B *being disjoint sets of literals, IRI nodes, and blank nodes as usual.*

We note that an RDF graph is a generalised graph, and the semantic notions of interpretations, satisfaction and entailment carry over without problem. We will use *graph* without qualifier to be a generalised graph.

We use the concept of interpretation from RDF semantics [4]. For an interpretation I, we will write V_I as the related vocabulary, and $(\mathrm{IR}_I, \mathrm{IP}_I, \mathrm{IEXT}_I, \mathrm{IS}_I, \mathrm{IL}_I, \mathrm{LV}_I)$ as the corresponding sextuple (we use LV as the literal subset as opposed to $\mathcal{LV}$ a lexical-to-value mapping).

We define the set of nodes, the vocabulary of a graph G, and a syntactic class extension function for a node $u \in \mathrm{nd}(G)$.

$$\mathrm{nd}(G) = \{s, p, o : (s, p, o) \in G\} \tag{4}$$

$$\mathrm{vocab}(G) = \mathrm{nd}(G) \cap (L \cup U) \tag{5}$$

$$\dot{\mathrm{CEXT}}(G, u) = \{v : (v, \texttt{type}, u) \in G\} \tag{6}$$

5 Literals and Datatypes

We take the normal set theoretic view and terminology of functions as sets of pairs.

We use the concepts of literals, plain literals, typed literals and datatype maps from RDF [4, 10]. $l\text{^^}a$ is a typed literal where a is the datatype IRI.

Specifically, we take L_{plain} as the set of all plain literals. These are self-denoting, meaning that each is its own associated value.

A datatype map D is a set of pairs (a, d) where a is a IRI and d is a datatype. Each datatype comes with a value space, $\mathcal{V}_d$, a lexical space $\mathcal{L}_d$ and a lexical-to-value mapping $\mathcal{LV}_d$.

To represent this simply we use:

$$\mathcal{L}_D = L_{\mathrm{plain}} \cup \bigcup_{(a,d)\in D} \{l\text{^^}a : l \in \mathcal{L}_d\} \tag{7}$$

$$\mathcal{V}_D = L_{\mathrm{plain}} \cup \bigcup_{(a,d)\in D} \mathcal{V}_d \tag{8}$$

$$\mathcal{LV}_D = \mathrm{id}_{L_{\mathrm{plain}}} \cup \bigcup_{(a,d)\in D} \mathcal{LV}_d \tag{9}$$

Given the syntactic emphasis of the Herbrand approach we use to construct interpretations we represent the structure of the datatype map as the following graph.

$$\begin{aligned} G_D = &\{(l, \texttt{type}, a) : (a, d) \in D, v \in \mathcal{V}_d, l = \mathcal{VL}(v)\} \\ &\cup \{(a, \texttt{type}, \texttt{Datatype}) : (a, d) \in D\} \\ &\cup \{(l, \texttt{type}, \texttt{Literal}\ ) : l \in L_{\mathrm{plain}}\} \end{aligned} \tag{10}$$

This graph is true for all D-interpretations in the sense that the function IL can be extended to cover vocab(G_D), and then the modified interpretation is still a D-interpretation and it satisfies G_D.

Ter Horst [5] omitted the following corollary to his lemma 4.10

Theorem 1. *For any datatype map D, not using RDF vocabulary, except* `XMLLiteral`*, the D-semantics is consistent.*

6 Well-formed RDF Graphs and Datatype Maps

Our main theorem does not apply to all RDF graphs. For example, {(`Thing`, `sameAs`, `Nothing`)} does not have an OWL Full$_c$ interpretation. In this section we specify the constraints, which are combined constraints on a datatype map and a graph, that must be satisfied.

Some of the constraints are syntactic, and are designed to prevent too many unexpected consequences under RDFS entailment for triples added during our construction. The omitted preliminary definitions define *declared* subproperties, subclasses, domains and ranges, by chasing `subPropOf` and `subClassOf` chains through the graph.

We will refer to the five properties: `subPropOf`, `range`, `domain`, `subClassOf` and `type` as the *protected properties*.

Definition 5. *A combination of a datatype map D and an RDF graph G is* well-formed *if all of the following hold:*

1. *There is a D-interpretation I of G that does not map any of the RDF, RDFS or OWL vocabulary to values in $\mathcal{V}_D$, and does not map any of these items, except* `XMLLiteral`*, to any of the datatypes in* ran(D).
2. *G contains no typed literals from datatypes not in D.*
3. *G contains no proper declared superproperties of the protected properties*
4. *None of the declared domains or ranges in G of the protected properties, or of* `first` *or* `rest` *are in* dom(D) $\cup$ {`Literal`}.
5. `nil` *is the subject of no triple in G which has predicate being a declared subproperty of* `first` *or* `rest`.
6. *If $(s,p,o),(s,p',o') \in G$, and for $q \in$ {`first`,`rest`}, if both p and p' are declared subproperties of q then $o = o'$.*
7. *G does not use the OWL vocabulary.*
8. *$\mathcal{LV}_D$ is surjective.*
9. *Neither* `rdfs:Class` *and* `Prop` *are proper declared superclasses of* `Resource` *or* `Prop`.

From the first and second points, G contains no ill-typed literals. The third and fourth point means that we can add triples for the protected properties during our construction without worrying about unexpected consequences from rules rdfs2, rdfs3 and rdfs7. The fourth, fifth and sixth points ensures that collections are well-behaved. The seventh reflects the modest ambition of this paper. The eighth holds for the usual datatype map from XML Schema. The ninth point avoids further difficulties.

7 Main Theorem

We can now state the main theorem[3]. We sketch the proof in the next several sections.

Theorem 2. *For a well-formed combination of an RDF graph G and a datatype map D there is an OWL Full$_c$ interpretation.*

Corollary 1. *OWL Full$_c$ is consistent.*

Proof. Take the usual datatype map D and G as the empty graph. This has a D-interpretation by the previous theorem, and so is well-formed.

Most 'real world' RDF graphs that do not use the OWL vocabulary are well-formed with the XML Schema datatype map, i.e. the theorem has practical importance!

8 Rules and Closures

We use six of the entailment rules from [4]. We refer to them as the set R, which contains rdfs2, rdfs3, rdfs7, rdfs9, rdfs12 and rdfs13. These deal with `domain`, `range`, `subPropOf`, `subClassOf`, `ContMemProp`, and `Datatype` respectively. Amongst the omitted rules are the rules for the transitivity and reflexivity of `subPropOf` and `subClassOf`. We use these rules without any of the side conditions specified in [4]. The action of each rule is that if the triples in the precondition match triples in a graph G, then we can construct a result triple, by substituting the variables.

We define closed and rule closure in the usual way:

Definition 6. *A graph G is closed with respect to a set of rules R, if for every rule $r \in R$, and every subset $m \subset G$ matching the precondition of r, the resulting consequent triple $r(m) \in G$.*

i.e. a graph G is R-closed if none of the rules when applied to G create new triples, not already in G. Since we are mainly interested in infinite graphs, we use an abstract definition of closure, which corresponds to exhaustive application in the finite case.

Definition 7. *Given a set of rules R, the R-closure of G, (written $R(G)$) is the minimal R-closed graph containing G.*

We omit the other entailment rules in [4], and so have to achieve the same effect in other ways.

- The rules lg and gl are addressed by using general graphs.
- The rules rdf1 and rdfs4a, rdfs4b, rdfs6 and rdfs10 are addressed by explicit initialization, see below.
- The rules rdf2, rdf2D and rdfs1 are addressed by G_D defined above.
- The rule rdfs5 is addressed using the "if and only if" semantics for `subPropOf`.
- The rules rdfs8 and rdfs11 are addressed using the "if and only if" semantics for `subClassOf`.

[3] This theorem is due to the first author.

8.1 Resources and Properties of a graph

To replace rdf1, we define:

$$\mathit{\Pi}(G) = \{(p, \texttt{type}, \texttt{Prop}\ ) : (s, p, o) \in G\} \tag{11}$$

Application of this rule after any of the rules in R is unnecessary, because any resulting triple has a predicate which is either explicitly listed as a `Prop` in the axiomatic triples, or as a consequence of the axiomatic triples and the preconditions of the rule.

To replace rdfs4a and rdfs4b we define:

$$\mathit{\Phi}(G) = \{(u, \texttt{type}, \texttt{Resource}\ ) : u \in \mathrm{nd}(G)\} \tag{12}$$

Application of this rule after any of the other rules is unnecessary because the only nodes that can be added are those in $\mathrm{nd}(G)$ which are all listed as `Resource`'s either explicitly in the axiomatic triples or implicitly so (by virtue of rule rdfs9).

9 Syntactic construction of an OWL Full$_c$ universe

To make a simple universe, or domain of discourse, we construct a graph whose nodes are in one-one correspondence with the elements of that universe. This means, in particular, that we have to simplify the representation of literals in the graph. All nodes interpreted as literals, whether in canonical form or not, whether syntactically literal, or IRI node or blank node, will be replaced by a canonical form (from $\mathcal{VL}_D$).

The construction comes in two phases. In the first phase, we start with the RDF graph G given in the main theorem, and add:

1. initialisation triples G_D for the datatype map
2. the RDF and RDFS axiomatic triples,
3. some trivial RDFS-consequences,
4. and take the R-closure.

The resulting graph is G_4 and is D-satisfiable. We then simplify the representation of all literals in this graph using a known D-interpretation and $\mathcal{VL}_D$. We use this simplified graph in the second stage. This graph still contains no OWL vocabulary.

In the second stage we build a Herbrand model. This is a graph that contains all the triples needed to explicitly represent all the properties in the universe. Unlike in Hayes' or ter Horst's work, we do not just add necessary consequences but also add contingent facts that happen to be true in this particular model, but are not necessarily true. For example, OWL requires a property `priorVers` to exist. In our universe like in many real life ontologies this property will be empty. Thus it will also be functional, inverse functional, transitive and symmetric. These four facts about our universe have to be recorded explicitly because of the "if and only if" conditions on the related classes. However, they are contingent in that in some other universe in which the initial graph is true, `priorVers` has a rich structure and has none of these properties.

9.1 Replacing the literals

This graph G_4 has potentially many different nodes that denote the same literal under a given D-interpretation I. To simplify the rest of the proof, we want to construct a new graph H_0 that does not have that property.

From the definition of satisfaction in [4] there is a specific interpretation $\mathcal{I}$, and a specific function $\mathcal{A}$ from the blank nodes of G to $\mathrm{IR}_{\mathcal{I}}$ such that $\mathcal{I} + \mathcal{A}$ satisfies G_4. We will use these symbols unchanged in later sections, referring back to the *same* functions used in the following constuction of H_0 using $G_4, \mathcal{I}, \mathcal{A}$ and $\mathcal{VL}_D$.

We use an auxiliary node mapping function ψ:

$$\psi : \mathrm{nd}(G_4) \to \mathrm{nd}(G_4) \cup \mathcal{L}_D \tag{13}$$

$$\psi(n) = \begin{cases} \mathcal{VL}((I + A)(n)) & \text{if } n \in \textsc{cext}(G_4, \texttt{Literal}), \\ n & \text{otherwise.} \end{cases} \tag{14}$$

$$H_0 = \{(\psi(s), \psi(p), \psi(o)) : (s, p, o) \in G_4\} \tag{15}$$

9.2 Phase 2: mechanics

The second stage consists of a number of steps, each with a related attribute of the graph. Each step adds triples to the graph in order that it satisfy the corresponding attribute. In the initial steps (1-5) of this stage, we deal with the 'if' conditions of RDFS. This is done in several pieces mainly so that we can use the reduced rule set R. The advantage of this over Hayes' or ter Horst's approach is that the proofs are easier. The later steps (6-12) deal with OWL vocabulary and the "... and only if" conditions on the RDFS vocabulary. The attribute of the graph is a syntactic rule based on the semantic conditions for the relevant triples of that step (typically the conditions for the predicate). The triples added do not negate the attributes of the graph achieved in earlier steps. Thus we gradually build a graph with more and more syntactic attributes corresponding to conditions from the semantics. The final graph has all the attributes and is in one-one correspondence with an OWL $\mathrm{Full_c}$ universe. We can then build a Herbrand interpretation in the usual way.

The initial steps of the second stage are as follows:

1. Add axiomatic and other triples for OWL $\mathrm{Full_c}$, (tables 2 and 3)
2. We take the R-closure.
3. Add some trivial RDFS consequences
4. Add additional triples that will be trivially true in our universe, for `domain`, `range`, `subPropOf` and `subClassOf`.
5. Take the R-closure again.

The triples generated in steps 6-12 are identified in table 1, with the addition that step 7 creates (? `type Datatype`) triples for finite classes of literals. d in steps 6-12 are identified in table 1, with the addition that step 7 creates (? `type`

`Datatype`) triples for finite classes of literals. We represent each of the steps as a function from graphs to graphs. The result of this function is then either a set of triples that need to be added, in the constructive step for that function, or that must already be present in the graph, for later steps.

With $\Gamma_1, \ldots \Gamma_{12}$ as defined below, we define, for $i = 1, \ldots 12$

$$H_i = H_{i-1} \cup \Gamma_i(H_{i-1}) \tag{16}$$

The following lemma expresses the fact that we keep moving forward in this process:

Lemma 1. *For each $i, j = 1, \ldots 12$, with $i \leq j$ and $i \neq 2$, we have $\Gamma_i(H_j) \subset H_j$.*

At the end of the process, H_{12} is such that it has an even stronger relationship with these functions, in that they precisely characterise the triples of various sorts that are present in H_{12}. The final function Γ_{12} is defined using an auxiliary function Ω. For $i = 6, \ldots 11$, each of the functions Γ_i, and also for $i = 12$, Ω, produces a graph consisting of triples matching the expressions in table 1, in which ? stands for any value. We take M_i to be the simple graph to graph function that selects exactly those triples that match the ith row of the table, so that, for example, $M_{10}(G) = \{(s, \texttt{inverseOf}, o) \in G\}$.

i	*Predicates or Triple Patterns*	
6	`sameAs differentFrom`	
7	(? `type FunctionalProp`)	(? `type InvFunProp`)
	(? `type SymProp`)	(? `type TransProp`)
8	`complmntOf subClassOf disjointWith equivClass`	
9	`unionOf intersectionOf oneOf distinctMembers`	
10	`inverseOf`	
11	`domain range`	
12	`subPropOf equivProp`	

Table 1. The definition of M_i

Lemma 2. *For $i = 6, \ldots 11$, $M_i(\Gamma_i(H_{12})) = M_i(H_{12})$, and $\Omega(H_{12}) = M_{12}(H_{12})$.*

Axiomatic and Other Triples. Axiomatic and additional triples are given in tables 2, 3. They include blank nodes from a set disjoint with $\mathrm{nd}(H_0)$. b_1 is the same blank node in both tables. No other blank nodes are introduced later. The table of other triples are assorted *ad hoc* facts about our particular universe and interpretation.

$\Gamma_1(G)$ is simply the graph formed by the union of these two tables.

RDFS related. $\Gamma_5(G) = \Gamma_2(G) = R(G)$ where R is our set of six RDFS rules.

$\Gamma_3(G) = \Pi(G) \cup \Phi(G)$, gives trivial consequences corresponding to rules rdf1, rdfs4a and rdfs4b of [4].

The definition of G_4 ensures that when we add `subClassOf`, `subPropOf`, `domain` and `range` triples in later steps, there are no new consequences from rules rdfs2 and rdfs3 in R, since these have already been added in step 5.

$$\begin{aligned}\Gamma_4(G) =& \{(p, \texttt{domain}, \texttt{Resource}), (p, \texttt{range}, \texttt{Resource}), (p, \texttt{subPropOf}, p) : \\ & p \in \dot{\text{CEXT}}(G, \texttt{Prop})\} \\ & \cup \{(\texttt{priorVers}, \texttt{domain}, c), (\texttt{priorVers}, \texttt{range}, c), \\ & (c, \texttt{subClassOf}, c) : c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class})\}\end{aligned} \tag{17}$$

These are plausible, because: the `domain` and `range` triples follow the OWL "if and only if' semantics for these properties, remembering the contingent fact that in our interpretation `priorVers` has empty property extension; the other triples are necessary RDFS consequences (from rules rdfs6 and rdfs10).

"...and only if" The definitions of $\Gamma_6, \ldots \Gamma_{12}$ are straightforward syntactic variants of the corresponding semantic conditions for the properties given in table 1, and are found in [6]. There are only two complications. The first is that Γ_7 also introduces triples of the form $(a, \texttt{type}, \texttt{Datatype})$ and $(a, \texttt{subClassOf}, \texttt{Literal})$ for all finite, nonempty classes whose class extension is a subset of the class extension of `Literal`. The second is that in Γ_{12} the property extension of `subPropOf` and `equivProp` are changed, and they may both become superproperties of other properties. Thus a closure is needed. Proving this closure works is one of the more complicated parts of the proof.

Proving lemmas 1 and 2 depends crucially on the ordering of the steps, in that the later steps do not undermine the work done in the earlier steps; and on the initialization tables, in that when the earlier steps may get things wrong, this is patched up with the additional triples. The proof of step 7 is particularly intricate. There are 56 different cases to consider, 18 of which depend on identifying relevant triples that are not in H_{12}, and 27 of which depend on identifying pairs of relevant triples that are in H_6. For example, `oneOf` is not symmetric because $(\texttt{nil}, \texttt{oneOf}, b_2) \notin H_6$ or H_{12}.

10 An OWL Full$_c$ interpretation of H_{12} and G_0

We define a simple interpretation $\mathcal{J}$ of $V_{\mathcal{J}} = \text{vocab}(H_{12}) \cup \text{vocab}(G_4)$, using the datatype map D, the D-interpretation $\mathcal{I}$ used in section 9.1, via the auxiliary node mapping function ψ used in that section.

We take the universe U as the nodes of H_{12} except those that represent literals or datatypes, and we define two auxiliary functions: χ, a one-one mapping

between $\mathrm{nd}(H_{12})$ and U; and θ, that extends the domain of χ to include $\mathrm{nd}(G_4)$

$$U = (\mathrm{nd}(H_{12}) \setminus (\mathrm{dom}(D) \cup \mathcal{L}_D)) \cup \mathcal{V}_D \cup \mathrm{ran}(D) \tag{18}$$

$$\chi : \mathrm{nd}(H_{12}) \to U \tag{19}$$

$$\chi(x) = \begin{cases} d & (x,d) \in D \\ \mathcal{LV}(d) & x \in \mathcal{L}_D \\ x & \text{otherwise} \end{cases} \tag{20}$$

$$\theta : \mathrm{nd}(H_{12}) \cup \mathrm{nd}(G_4) \to U \tag{21}$$

$$\theta(x) = \begin{cases} \chi(x) & x \in \mathrm{nd}(H_{12}) \\ \chi(\psi(x)) & x \in \mathrm{nd}(G_4) \end{cases} \tag{22}$$

(22) is well-defined, because if $x \in \mathrm{nd}(H_{12} \cap \mathrm{nd}(G_4))$ then $x = \psi(x)$.

The interpretation $\mathcal{J}$ is then defined as:

$$\mathrm{IR}_{\mathcal{J}} = U \qquad \mathrm{IS}_{\mathcal{J}}(x) = \theta(x) \qquad \mathrm{IL}_{\mathcal{J}}(x) = \theta(x) \tag{23}$$

$$\mathrm{IP}_{\mathcal{J}} = \{\chi(p) : p \in \dot{\mathrm{CEXT}}(H_{12}, \texttt{Prop})\} \tag{24}$$

$$\mathrm{IEXT}_{\mathcal{J}}(x) = \{(\chi(s), \chi(o)) : (s, \chi^{-1}(x), o) \in H_{12}\} \tag{25}$$

$$\mathrm{LV}_{\mathcal{J}} = \{\chi(l) : l \in \dot{\mathrm{CEXT}}(H_{12}, \texttt{Literal})\} \tag{26}$$

The definitions of $\mathrm{IEXT}_{\mathcal{J}}, \mathrm{IS}_{\mathcal{J}}, \mathrm{IL}_{\mathcal{J}}$ are restricted to their respective domains of $\mathrm{IP}_{\mathcal{J}}$, IRIs and typed literals in $V_{\mathcal{J}}$.

Lemma 5. *$\mathcal{J}$ satisfies H_{12} and G_4.*

10.1 $\mathcal{J}$ is an OWL Full$_c$ interpretation

After establishing, in the omitted lemmas 6, 7, and 8, that $\mathcal{J}$ is a D-interpretation, we then consider each of the conditions from [1], sections 5.2 and 5.3, along with the additional modifications made in our section 3, in order to prove that it is an OWL Full$_c$ interpretation. Most of these follow directly from lemma 2, or from the axiomatic triples.

The most straightforward part is to check the constraints concerning restrictions. Since none of the relevant properties have non-empty property extension under $\mathcal{J}$, there is nothing to prove.

11 OWL without Comprehension

Having discarded the comprehension principles, we need to articulate what to do without them.

We start at the application level. The comprehension principles play no part in application operation. Applications from the RDF, RDFS world-view, concentrate largely on A-box consequences, and the comprehension principles all

concern the T-box. This reflects the emphasis of the description logic community, prior to OWL, on T-box reasoning, and fitted unnaturally with RDF. Even applications from the description logic viewpoint do not implicitly use the comprehension principles, when one rearticulates their operation in terms of OWL Full. For example, a classifier that given an ontology, provides a picture of the class hierarchy, generally show only those classes, and possibly class expressions, that are already written down within the ontology. A few may show a few additional class intersections or unions, which would require a finite and measured introduction of unnamed classes currently articulated in OWL Full using the infinite and unmeasured comprehension principles. A further style of description logic application that could be seen as dependent on the comprehension principles would be a query answering system that allows the user to type in a class expression and ask whether it is empty or not. Such a question presupposes that the class expression is not itself simply 'false'.

At the architectural level, the comprehension principles play a vital rôle in theorem 2 of [1]. This articulates the relationship between OWL DL and OWL Full, on the DL syntactic subset specified by the mapping rules. It is unfortunately defective in giving only a one-sided implication, and lacking an articulation of when the converse implication fails. So it does not give an account of similarities and differences between OWL DL and OWL Full, which would have been a more useful result. Such an account is impossible to give before showing that OWL Full is consistent. Our goal for a reworked theorem 2 would be to change OWL Full such that a consistency proof is possible (or hopefully even easy!) and then have a theorem 2 that gives a full and precise account of the relationship between the direct semantics and the OWL Full semantics on the DL syntactic subset.

Given that the direct semantics for OWL, follows classical logic, and finds implications such as (3) as entailments, and OWL Full_c does not, a comprehension rule is required in such an articulation of the relationship. A possible approach, would be to replace the comprehension principles, that operate on the semantic level, with syntactic *comprehension rules*. Each of these would be such that, given nodes that exist in a graph (or in the vocabularies being used) it would produce a syntactic representation of the resources currently required by the comprehension principles. Then a *comprehension sequence* from a graph G_0 to G_n would consist of a finite sequence of rule applications to add more and more such terms. We conjecture that appropriate rules could be found so that as long as all the G_i are in the DL syntactic subset, such a sequence would preserve OWL Full_c satisfiability. If this is indeed so, for the purposes both of stating and proving a revamped theorem 2, one could define a notion of generalized entailment from G to H, being that there is a comprehension sequence within the DL subset from $G = G_0$ to G_n, such that G_n OWL Full_c entails H. This notion would also suffice to articulate within OWL Full_c the finite uses of comprehension we saw in a few OWL DL style applications.

12 Conclusions

12.1 OWL Full is too complicated

The most striking thing about proving OWL Full_c consistent is that it is unreasonably hard work. It is a modest goal but would take two or more papers to do it justice. We do not believe that a consistency proof for OWL Full will be achieved any time soon. In fact, we doubt whether OWL Full is consistent.

The requirement that OWL Full does not break RDF and RDFS as expressed in our main theorem, is entirely reasonable. It seems like an architectural principle that should have been included in the RDF Semantics document: "Semantic extensions MUST NOT ... [make consistent graphs that do not use their vocabulary inconsistent]".

Ter Horst [11] shows that a large number of "if ... then" conditions can be accommodated as rules with very little problem. The parts of the OWL Full_c semantics that were either based on "if ... then" or were based on constructs such as restrictions that simply do not occur in RDF except by using the OWL vocabulary were easy; trivial even, for our minimalist goal.

The problem we solved were the cyclic dependencies introduced by the "if and only if" conditions. These are at their worst for `subPropertyOf`, where the question of which property is a subproperty of `subPropertyOf` is hard to duck, and hard to answer. But there are many other potential circularities that are addressed in our construction like 'is `inverseOf` functional?' versus 'does `type` have an inverse?'

While we do not advocate dropping all "if and only if"s, each one is costly and needs justification rooted in actual practice: running code that is best understood with a semantics that has that particular condition. Consider the classifier that displays a class hierarchy. Each of the lines on the display corresponds to a `subClassOf` relationship that has been inferred using "if and only if" semantics. We do not believe that any tool supports some such feature for `inverseOf`, `subPropertyOf`, `range`, `unionOf` .... If there are none or even only a few such tools, then the best decision is to remove those conditions from the semantics of OWL Full.

The reason for our concern is the rôle of OWL Full as the lynch pin that holds together RDF (cheap and cheerful, scalable, with wide deployment) and OWL DL (heavyweight, industrial strength reasoning). If these two diverge too much then both communities will lose the benefits that our complementary strengths bring. Schneider's result shows that some reworking of OWL1 Full is needed before OWL2 proceeds past Candidate Recommendation. This presents an opportunity to also fix several of the more severe non-critical foundational issues both with OWL Full and with RDF: the excess of "if and only if" conditions; generalizing the RDF abstract syntax [9]; named graph support. While only comprehension needs work in order to complete OWL2, all of the others would help OWL2 and be wider improvements for the RDF and OWL1 communities.

12.2 Summary

This paper has shown that OWL Full without the comprehension principles is self-consistent and more strongly is consistent with the D-semantics for most non-OWL RDF graphs. We have in the process pointed to several weaknesses of the OWL Full semantics most notably but by no means exclusively the comprehension principles themselves.

Because of the Schneider paradox OWL Full semantics need some redesign. This paper has shown that a possible route is to convert the comprehension principles into syntactic devices. We advocate a more radical overhaul.

Returning to the subject of the wart many of us care so much more about the DL left-side profile of OWL that we feel we can ignore the wart on the Full right-side. We close by remembering that the Semantic Web is seen by most people outside our relatively small community, face on with both the right and the left sides showing. Fixing the difficult problem of OWL Full consistency would benefit all of us.

References

1. Patel-Schneider, P.F., Horrocks, I., Hayes, P.: OWL Web Ontology Language Semantics and Abstract Syntax. W3C recommendation, W3C (February 2004) http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.
2. Motik, B., Patel-Schneider, P.F., Horrocks, I.: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Working Draft, W3C (2008) http://www.w3.org/TR/2008/WD-owl2-syntax-20080411/.
3. Carroll, J.J.: An OWL Full Interpretation. Technical Report, HP Labs (2008) HPL-2008-60.
4. Hayes, P.: RDF Semantics. W3C recommendation, W3C (February 2004) http://www.w3.org//TR/2004/REC-rdf-mt-20040210/.
5. ter Horst, H.J.: Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. J. Web Semantics **3**(2-3) (2005) 79–115
6. Carroll, J.J., Turner, D.: The Consistency of OWL Full (with proofs). Technical Report, HP Labs (2008) HPL-2008-59.
7. Patel-Schneider, P.F., Fensel, D.: Layering the Semantic Web: Problems and Directions. In Horrocks, I., Hendler, J., eds.: Proc. of the 1st International Semantic Web Conference (ISWC2002). (2002)
8. Schneider, M.: OWL 2 Full: Current State and Issues (2008) http://lists.w3.org/Archives/Public/public-owl-wg/2008Apr/0029.
9. de Bruijn, J.: RIF RDF and OWL Compatibility. W3C Working Draft, W3C (2008) http://www.w3.org/TR/2008/WD-rif-rdf-owl-20080415/.
10. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, W3C (February 2004) http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.
11. ter Horst, H.J.: Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In: International Semantic Web Conference. (2005) 668–684

(owl:Class type rdfs:Class)	(priorVers type OntProp)	(Nothing complmntOf Resource)
(rdfs:Class subClassOf owl:Class)	(backCompat type OntProp)	(subClassOf type TransProp)
(Prop subClassOf ObjProp)	(incompat type OntProp)	(AnnProp subClassOf Prop)
(DataProp subClassOf Prop)	(disjointWith type Prop)	(OntProp subClassOf Prop)
(b_1 type AllDifferent)	(inverseOf type Prop)	(Prop subClassOf Thing)
(Resource subClassOf Thing)	(differentFrom type Prop)	(Resource disjointWith Nothing)
(Nothing type rdfs:Class)	(complmntOf type Prop)	(Nothing disjointWith Nothing)
(DeprClass type rdfs:Class)	(unionOf type Prop)	(Nothing disjointWith Resource)
(DeprProp type rdfs:Class)	(intersectionOf type Prop)	(Thing equivClass Resource)
(Restrict subClassOf rdfs:Class)	(oneOf type Prop)	(Thing equivClass Thing)
(Literal type Datatype)	(allValuesFrom type Prop)	(Resource equivClass Resource)
(Datatype subClassOf rdfs:Class)	(onProperty type Prop)	(Resource equivClass Thing)
(subClassOf type TransProp)	(someValuesFrom type Prop)	(DataRange type rdfs:Class)
(subPropOf type TransProp)	(hasValue type Prop)	(Thing sameAs Thing)
(versionInfo type AnnProp)	(minCardinality type Prop)	(Thing differentFrom Nothing)
(label type AnnProp)	(maxCardinality type Prop)	(inverseOf inverseOf inverseOf)
(comment type AnnProp)	(cardinality type Prop)	(inverseOf subPropOf inverseOf)
(seeAlso type AnnProp)	(distinctMembers type Prop)	(equivProp type SymProp)
(isDefinedBy type AnnProp)	(Thing complmntOf Nothing)	(equivProp inverseOf equivProp)
(Ontology type rdfs:Class)	(Resource complmntOf Nothing)	
(imports type OntProp)	(Nothing complmntOf Thing)	

Table 2. The axiomatic triples for OWL Full$_c$

(inverseOf type SymProp)	(rdfs:Class type b_6)	(b_{12} inverseOf b_9)
(priorVers inverseOf priorVers)	(rdfs:Class type b_7)	(b_8 b_8 b_2)
(priorVers inverseOf backCompat)	(rdfs:Class unionOf b_2)	(b_8 b_8 b_3)
(incompat inverseOf backCompat)	(owl:Class unionOf b_2)	(b_3 b_8 b_2)
(imports inverseOf priorVers)	(rdfs:Class unionOf b_3)	(b_2 b_8 b_4)
(priorVers equivProp priorVers)	(b_2 unionOf nil)	(b_2 b_9 b_3)
(priorVers equivProp backCompat)	(rdfs:Class intersectionOf b_2)	(b_4 b_{10} b_5)
(incompat equivProp backCompat)	(owl:Class intersectionOf b_2)	(b_2 b_{11} b_3)
(priorVers type FunctionalProp)	(rdfs:Class intersectionOf b_3)	(b_3 b_{12} b_2)
(priorVers type InvFunProp)	(b_3 intersectionOf b_5)	(b_2 type b_8)
(b_2 type rdfs:Class)	(b_7 oneOf b_3)	(b_2 type b_9)
(b_2 first rdfs:Class)	(b_3 oneOf b_2)	(b_3 type b_9)
(b_2 rest nil)	(b_2 oneOf nil)	(b_{12} domain b_{10})
(b_3 first rdfs:Class)	(b_6 oneOf b_2)	(b_9 domain b_{10})
(b_3 rest nil)	(b_6 oneOf b_3)	(b_{10} domain b_8)
(b_4 first Nothing)	(b_1 distinctMembers b_2)	(b_9 range b_{10})
(b_4 rest nil)	(b_1 distinctMembers b_3)	(b_{10} range b_8)
(b_5 first b_3)	(b_8 subPropOf b_8)	(priorVers subPropOf b_9)
(b_5 rest nil)	(b_8 equivProp b_8)	
(rdfs:Class type b_3)	(b_9 inverseOf b_{12})	

Table 3. Additional triples assumed in the proof.