The Consistency of OWL Full (with proofs)

Jeremy J. Carroll, Dave Turner
HP Laboratories
HPL-2008-59
May 21, 2008*

OWL, semantics, Herbrand, OWL Full

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. This longer version contains the proofs.

External Accession Date Only

Approved for External Publication

The Consistency of OWL Full (with proofs)

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. This longer version contains the proofs.

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 final two sections of the paper.

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 Prerequisites

While other readers may gain something from this paper, the target audience for the technical material is familiar with RDF Semantics [4], the RDFS compatible semantics for OWL (OWL Full) [1], and to a lesser extent, ter Horst's work on D* semantics [5].

The technical details of this paper are difficult. A possible first reading is to skip over everything to do with datatypes, literals, `AllDifferent` and `distinctMembers`, and then to return to these details once the main structure of the proof is understood.

1.3 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, these being: `Prop`*erty*, `subProp`*erty*`Of`, `equiv`*alent*`Class`, `priorVers`*ion*, `back`*ward*`Comp`*atible*`With`, `incompat`*ible*`With`,

FunctionalProperty, InverseFunctionalProperty, SymmetricProperty, TransitivieProperty, ObjectProperty, AnnotationProperty, OntologyProperty, DeprecatedProperty, DeprecatedClass, Restriction, complementOf and ContainerMembershipProperty.

1.4 Outline of Paper

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.

There is a short version [6] available. It omits most of the technicalities.

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.

```
eg:C owl:unionOf _:c1
_:c1 rdf:first eg:A
_:c1 rdf:rest _:c2
_:c2 rdf:first eg:B
_:c2 rdf:rest rdf:nil
```

The problem is that the $A \cup B$ is written in OWL with a list as shown, 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. In OWL2, this would be expressed very similarly to [7] as:

```
_:1 type Restrict
_:1 onProperty type
_:1  maxQualifiedCardinality 0
_:1  onClass _:2
_:2 oneOf _:3
_:3 rdf:first _:1
_:3 rdf:rest rdf:nil
```

In OWL1, this can be articulated as follows:

```
_:1 complmntOf _:2
_:2 type Restrict
_:2 onProperty type
_:2 someValuesFrom _:3
_:3 oneOf _:4
_:4 rdf:first _:1
_:4 rdf:rest rdf:nil
```

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.

```
_:1 complmntOf _:2
_:2 type SelfRestriction
_:2 onProperty type
```

While a a comprehension principle for self-restrictions could have an *ad hoc* exclusion for `type`, eliminating this particular case, there is no overall design criterion that would allow any sense of plausibility to a claim of consistency, or utility, of such a principle without a proof.

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 below.

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

3.1 $n \geq 1$

The first of the "Further conditions on *owl:oneOf*" should have an additional constraint of $n \geq 1$. Otherwise (`Nothing type Datatype`) follows.

3.2 Treatment of `distinctMembers`

`distinctMembers` is underspecified, and what specification that there is, is in the comprehension principle.

An alternative formulation used here, is:

- IAD $\neq \emptyset$
- If $E =$ `distinctMembers` then $(x, y) \in \text{EXT}_I(S_I(E))$ if and only if $\exists x \in C_I$, and y is a sequence of $y_1, \ldots, y_n$ over IOT with $y_i \neq y_j$ for $1 \leq i < j \leq n$.

The first replaces the comprehension part of the comprehension principle, but only requires a single item; the second echos the definition of other list-valued properties such as `unionOf`, combined with (non-comprehension) parts of the comprehension principle for `distinctMembers`.

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. If we wish to restrict ourselves to an RDF graph we will say 'RDF 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 $(\text{IR}_I, \text{IP}_I, \text{IEXT}_I, \text{IS}_I, \text{IL}_I, \text{LV}_I)$ 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 also syntactic class and property extension functions, for a node $u \in \text{nd}(G)$.

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

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

$$\text{CE}\dot{\text{X}}\text{T}(G, u) = \{v : (v, \texttt{type}, u) \in G\} \tag{6}$$

$$\text{IE}\dot{\text{X}}\text{T}(G, u) = \{(s, o) : (s, u, o) \in G\} \tag{7}$$

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. This is merely a syntactic expression of a simple pair.

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:

$$\mathcal{L}_D = L_{\text{plain}} \cup \bigcup_{(a,d)\in D} \{l^{\wedge\wedge}a : l \in \mathcal{L}_d\} \tag{8}$$

$$\mathcal{V}_D = L_{\text{plain}} \cup \bigcup_{(a,d)\in D} \mathcal{V}_d \tag{9}$$

$$\mathcal{LV}_D = \text{id}_{L_{\text{plain}}} \cup \bigcup_{(a,d)\in D} \mathcal{LV}_d \tag{10}$$

We assume that $\mathcal{LV}_D$ is surjective[3].

We also choose a function that picks out arbitrary 'canonical' lexical representatives for each value, i.e. we take[4] a function $\mathcal{VL}_D$ that is a subset of $\mathcal{LV}_D^{-1}$.

The standard datatype map derived from XML Schema is complicated and has a rich structure, (see [12, 13]). This structure needs to be represented in an OWL Full$_c$ interpretation. 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.

$$\begin{aligned} G_D = &\{(l, \texttt{type}, a) : (a, d) \in D, v \in \mathcal{V}_d, l = \mathcal{VL}(v)\} \\ &\cup \{(a, \texttt{type}, \texttt{Datatype}) : (a, d) \in D\} \\ &\cup \{(l, \texttt{type}, \texttt{Literal}\ ) : l \in L_{\text{plain}}\} \end{aligned} \tag{11}$$

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.

5.1 D* interpretations

Herman ter Horst [5] greatly extended Pat Hayes [4] method of using Herbrand closures to investigate semantic properties of semantic extensions to RDF. He makes the systematic choice to have a one-sided semantics, with only "if" conditions, rather than the "if and only if' conditions of OWL Full.

[3] This may not be the case if the map contains XML Schema [11] union datatypes but not the types from which they are derived. The standard datatype maps suggested for RDF and OWL are surjective. This restriction limits the proof to countable datatypes. Alternatively, the lexical forms could allow transfinite sequences of characters. Countable sequences would suffice for the proposed `owl:real`.

[4] This does not need the axiom of choice in the countable case.

So his D* semantics, is identical to Pat Hayes' D semantics, except that instead of the equality: $\text{CEXT}_{\mathcal{J}}(d) = \mathcal{V}_d \subset \text{LV}_{\mathcal{J}}$ the weaker one-sided relationship $\text{CEXT}_{\mathcal{J}}(d) \supset \mathcal{V}_d$ is used, (where d is a datatype).

The construction in this paper, essentially follows ter Horst's method, but extends it with great care, to meet enough "if and only if"s to provide an OWL Full_c interpretation.

One key lemma in his paper is lemma 4.10. Unfortunately, it is couched in the technicalities of his paper, (partly because he is concerned with complexity results that are not of interest here) so we will paraphrase it as, any RDF graph G that does not contain an ill-typed literal, has a D* interpretation, specifically, being the natural interpretation of the Herbrand closure of G under ter Horst's rule set (taken from [4]).

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.*

Proof. Take the D* Herbrand closure of G_D as above. This does not contain a D-clash and so has a D* interpretation $S(G_D)$ as specified by ter Horst. Analysis of the class extension of the datatypes in D in this particular interpretation, shows that $\text{CEXT}_{S(G_D)}(d) = \mathcal{V}_d$ for every $(a, d) \in D$.

6 Well-formed RDF Graphs and Datatype Maps

Our main theorem does not apply to all RDF graphs. For example, $\{(\texttt{Thing}, \texttt{sameAs}, \texttt{Nothing})\}$ 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.

Definition 2. *For a graph G, and two nodes $p, q \in \text{nd}(G)$, p is a* declared subproperty *of q, and q is a* declared superproperty *of p, if and only if one of $p = q$ or there are nodes $r, s \in \text{nd}(G)$, with:*

1. *$(p, s, r) \in G$ (or in the axiomatic triples for RDF and RDFS)*
2. *s a declared subproperty of* `subPropOf` *in G*
3. *r a declared subproperty of q in G.*

So that for IRI nodes, if p is a declared subproperty of q in G, then G RDFS entails $(p, \texttt{subPropOf}, q)$. (Such an entailment holds, but is trivial for blank nodes).

Definition 3. *For a graph G, and two nodes $c, d \in \text{nd}(G)$, c is a* declared subclass *of d, and d is a* declared superclass *of c, if and only if one of $c = d$ or there are nodes $e, s \in \text{nd}(G)$, with:*

1. $(c, s, e) \in G$ *(or in the axiomatic triples for RDF and RDFS)*
2. s *a declared subproperty of* `subClassOf` *in* G
3. e *a declared subclass of* d *in* G.

Definition 4. *For a graph* G*, and two nodes* $p, c \in \mathrm{nd}(G)$*,* c *is a* declared range *of* p*, if and only if there is some declared superproperty* p' *of* p *in* G*, and some declared subclass* c' *of* c*, and some declared subproperty* r *of* `range`*, such that* $(p', r, c') \in G$*. A* declared domain *is similarly defined.*

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* $\mathrm{ran}(D)$.
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* $\mathrm{dom}(D) \cup \{\texttt{Literal}\}$.
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* $q \in \{\texttt{first}, \texttt{rest}\}$*, 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.*

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.

7 Main Theorem

We can now state the main theorem[5]. The proof is 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!

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

8 Rules and Closures

Table 1 gives some of the RDFS entailment rules from [4]. We will refer to this set of eleven rules as R.

	Preconditions	*Consequent*
rdf1	u p v	p `type Prop`
rdfs2	p `domain` u v p u	v `type` u
rdfs3	p `range` u v p u	w `type` u
rdfs4a	u p v	u `type Resource`
rdfs4b	u p v	v `type Resource`
rdfs6	u `type Prop`	u `subPropOf` u
rdfs7	p `subPropOf` q v p u	v q u
rdfs9	v `subClassOf` w u `type` v	u `type` w
rdfs10	u `type rdfs:Class`	u `subClassOf` u
rdfs12	v `type ContMemProp`	v `subPropOf member`
rdfs13	v `type Datatype`	v `subClassOf Literal`

Table 1. Some RDFS entailment rules

In [4, 5], these are used with various side conditions, which complicates things. With generalized graphs, we can avoid the side conditions, and 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. These rules are used in conjunction with the usual two infinite sets of axiomatic triples. We use $\mathrm{nd}(R)$ to be the set of nodes, other than variables, in the consequences of the rules.

These simple rules cannot add new nodes to a graph other than ones that are explicit in the rules themselves.

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.*

Formally:

$$R(G) = \bigcap \{G' : G' \subset (\mathrm{nd}(G) \cup \mathrm{nd}(R))^3, G \subset G', G' \text{ is } R\text{-closed}\} \quad (12)$$

We are, going to use the set R of the six RDFS rules from table 1. 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 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`.

9 Syntactic construction of an OWL Full$_c$ universe

OWL Full$_c$ is more complicated than pD*, and we deviate in a number of details from ter Horst's approach. Hence, we wish to have a simpler syntactic representation of the universe for our interpretation, to simplify proving that it is an OWL Full$_c$ interpretation. 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. and take the R-closure.

The resulting graph is G_3 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.

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 and 2) of this stage, we deal with the 'if' conditions of RDFS. The later steps (3-9) 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 Full_c universe. We can then build a Herbrand interpretation in the usual way.

The steps of the second stage are as follows, where steps 3, and 5 to 9, add triples for the specified predicates.

1. Add axiomatic and other triples for OWL Full_c, (tables 7 and 8)
2. We take the $R \cup R'$-closure, where R' is the set of additional rules in table 2. These rules add triples for for `domain`, `range`, for which OWL Full_c has "if and only if" semantics. Although the rules are unsound in general, they will be trivially true in our universe,
3. `sameAs`, `differentFrom`.
4. Adding `type` triples for the class extension of `FunctionalProp`, `InvFunProp`, `SymProp`, `TransProp`, and `Datatype`.
5. `complmntOf`, `subClassOf`, `disjointWith`, `equivClass`.
6. `unionOf`, `intersectionOf`, `oneOf` and `distinctMembers`
7. `inverseOf`
8. `domain`, `range`
9. `subPropOf`, `equivProp`

Preconditions	*Consequent*
odr1 p `type Prop`	p `range Resource`
odr2 p `type Prop`	p `domain Resource`
odr3 c `type rdfs:Class priorVers range` c	
odr4 c `type rdfs:Class priorVers domain` c	

Table 2. Entailment rules for `domain` and `range`

9.1 Taking the *R*-closure

We take $G_0 = G$, the given well-formed RDF graph from the statement of theorem 2.

We then form $G_1 = G_0 \cup G_D$ and by the observation at the end of section 5 this graph has a D-interpretation.

We take: G_2 as the union of G_1 with the RDF and RDFS-axiomatic triples from [4]. Since these are necessarily true for any D-interpretation, G_2 is D-satisfiable.

We form $G_3 = R(G_2)$, this too is D-satisfiable, since the rules in R are sound.

9.2 Replacing the literals

This graph G_3 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_3. We will use these symbols unchanged in later sections, referring back to the *same* functions used in the following constuction of H_0 using $G_3, \mathcal{I}, \mathcal{A}$ and $\mathcal{VL}_D$.

We use an auxiliary node mapping function ψ:

$$\psi : \mathrm{nd}(G_3) \to \mathrm{nd}(G_3) \cup \mathcal{L}_D \tag{13}$$

$$\psi(n) = \begin{cases} \mathcal{VL}((I + A)(n)) & \text{if } n \in \textsc{cext}(G_3, \texttt{Literal}), \\ n & \text{otherwise.} \end{cases} \tag{14}$$

$$H_0 = \{(\psi(s), \psi(p), \psi(o)) : (s, p, o) \in G_3\} \tag{15}$$

9.3 Phase 2: mechanics

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 $\Gamma_1, \ldots \Gamma_9$ as defined below, we define, for $i = 1, \ldots 9$

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

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

Lemma 1. *For each $i, j = 1, \ldots 9$, with $i \leq j$, we have $\Gamma_i(H_j) \subset H_j$.*

At the end of the process, H_9 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_9. The final function Γ_9 is defined using an auxiliary function Ω. For $i = 3, \ldots 8$, each of the functions Γ_i, and also for $i = 9$, Ω, produces a graph consisting of triples matching the expressions in table 9.3, 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, $M_7(G) = \{(s, \texttt{inverseOf}, o) \in G\}$.

i	*Patterns*	
3	(? `sameAs` ?)	(? `differentFrom` ?)
4	(? `type FunctionalProp`)	(? `type InvFunProp`)
	(? `type SymProp`)	(? `type TransProp`)
5	(? `complmntOf` ?)	(? `subClassOf` ?)
	(? `disjointWith` ?)	(? `equivClass` ?)
6	(? `unionOf` ?)	(? `intersectionOf` ?)
	(? `oneOf` ?)	(? `distinctMembers` ?)
7	(? `inverseOf` ?)	
8	(? `domain` ?)	(? `range` ?)
9	(? `subPropOf` ?)	(? `equivProp` ?)

Table 3. The definition of M_i

Lemma 2. *For* $i = 3, \ldots 8$, $M_i(\Gamma_i(H_9)) = M_i(H_9)$, *and* $\Omega(H_9) = M_9(H_9)$.

We defer the proofs of both lemmas until after we have defined the Γ_i.

We will then construct a combined OWL Full$_c$ interpretation of H_9 and G_3, using H_9 as the domain of discourse, completing the proof of the main theorem.

Axiomatic and Other Triples. Axiomatic and additional triples are given in tables 7, 8. We will refer to these as T_{ax} and T_{other}. They include blank nodes from a set disjoint with $\text{nd}(H_0)$. 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.

$\Gamma_1(G) = T_{\text{ax}} \cup T_{\text{other}}$ is simply the graph formed by the union of these two tables.

RDFS related. $\Gamma_2(G) = (R \cup R')(G)$ where R is our set of eleven RDFS rules in table 1, and R' is the set of four rules in table 2. The rules rdfs6, rdfs10 in R and the rules in R' ensure 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 2. The rules in R' 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.

"... and only if"

$$\begin{aligned}\Gamma_3(G) =& \{(u, \texttt{sameAs}, u) : u \in \text{nd}(G)\} \\ & \cup \{(u, \texttt{differentFrom}, v) : u, v \in \text{nd}(G), u \neq v\}\end{aligned} \tag{17}$$

$$
\begin{aligned}
\Gamma_4(G) = &\{(p, \texttt{type}, \texttt{FunctionalProp}) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}),\ \text{s.t.} \\
&\qquad \forall u, v_1, v_2 \in \text{nd}(G), (u, p, v_1), (u, p, v_2) \in G \Rightarrow v_1 = v_2\} \\
&\cup \{(p, \texttt{type}, \texttt{InvFunProp}) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}),\ \text{s.t.} \\
&\qquad \forall u, v_1, v_2 \in \text{nd}(G), (v_1, p, u), (v_2, p, u) \in G \Rightarrow v_1 = v_2\} \\
&\cup \{(p, \texttt{type}, \texttt{TransProp}) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}),\ \text{s.t.} \\
&\qquad \forall u, v, w \in \text{nd}(G), (u, p, v), (v, p, w) \in G \Rightarrow (u, p, w) \in G\} \\
&\cup \{(p, \texttt{type}, \texttt{SymProp}) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}),\ \text{s.t.} \\
&\qquad \forall u, v \in \text{nd}(G), (u, p, v) \in G \Rightarrow (v, p, u) \in G\} \\
&\cup \{(d, \texttt{type}, \texttt{Datatype}), (d, \texttt{subClassOf}, \texttt{Literal}) : d \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}), \\
&\qquad 1 \leq |\ \dot{\text{CEXT}}(G, d)\ | < \infty, \dot{\text{CEXT}}(G, d) \subset \dot{\text{CEXT}}(G, \texttt{Literal})\}
\end{aligned}
\tag{18}
$$

We will add no further type triples, so we can fix properties that relate classes. These properties depend on the class extension of various classes c, for which we use $\dot{\text{CEXT}}(G, c)$, which looks at the `type` triples in G.

$$
\begin{aligned}
\Gamma_8(G) = &\{c_1, \texttt{complmntOf}, c_2) : c_1, c_2 \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}), \\
&\qquad \dot{\text{CEXT}}(G, c_1) \cup \dot{\text{CEXT}}(G, c_2) = \dot{\text{CEXT}}(G, \texttt{Resource}), \\
&\qquad \dot{\text{CEXT}}(G, c_1) \cap \dot{\text{CEXT}}(G, c_2) = \emptyset\} \\
&\cup \{c_1, \texttt{disjointWith}, c_2) : c_1, c_2 \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}), \\
&\qquad \dot{\text{CEXT}}(G, c_1) \cap \dot{\text{CEXT}}(G, c_2) = \emptyset\} \\
&\cup \{c_1, \texttt{subClassOf}, c_2) : c_1, c_2 \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}), \\
&\qquad \dot{\text{CEXT}}(G, c_1) \subset \dot{\text{CEXT}}(G, c_2)\} \\
&\cup \{c_1, \texttt{equivClass}, c_2) : c_1, c_2 \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}), \\
&\qquad \dot{\text{CEXT}}(G, c_1) = \dot{\text{CEXT}}(G, c_2)\}
\end{aligned}
\tag{19}
$$

$\dot{\text{CEXT}}(H_0, \texttt{List})$ is not empty, containing `nil`, and several blank nodes from the additional triples, (table 8). There are also a number of empty classes, `Nothing` and `DeprClass`, for example. There may be other lists that were already present in G_0.

We use the phrase s is a sequence in G of $c_1, c_2, \ldots c_n$, as in [1] to mean that $s \in \dot{\text{CEXT}}(G, \texttt{List})$ and there exist nodes $c_1, c_2, \ldots c_n \in \text{nd}(G)$, with the appropriate `first` and `rest` triples, starting at s, and ending at `nil` are also in G.

We note that the definition 5 of well-formed graph, ensures that for each sequence s there is no ambiguity about which nodes the c_i are.

$$
\begin{aligned}
\Gamma_6(G) = &\{(c, \texttt{intersectionOf}, s) : c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}),\\
&\qquad s \text{ a sequence in } G \text{ of } c_1, \dots c_n \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class})\\
&\qquad \dot{\text{CEXT}}(G, c) = \bigcap_{i=1}^{n} \dot{\text{CEXT}}(G, c_i)\}\\
&\cup \{(c, \texttt{unionOf}, s) : c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}),\\
&\qquad s \text{ a sequence in } G \text{ of } c_1, \dots c_n \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class})\\
&\qquad \dot{\text{CEXT}}(G, c) = \bigcup_{i=1}^{n} \dot{\text{CEXT}}(G, c_i)\}\\
&\cup \left\{ \begin{array}{c} c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class}),\\ (c, \texttt{oneOf}, s) : s \text{ a sequence in } G \text{ of } a_1, \dots a_n \\ \dot{\text{CEXT}}(G, c) = \{a_1 \dots a_n\} \end{array} \right\}\\
&\cup \{(a, \texttt{distinctMembers}, s) : a \in \dot{\text{CEXT}}(G, \texttt{AllDifferent}),\\
&\qquad s \text{ a sequence in } G \text{ of } a_1, \dots a_n\\
&\qquad \forall 1 \le i < j \le n, a_i \ne a_j\}
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
\Gamma_7(G) = &\{(p_1, \texttt{inverseOf}, p_2) : p_1, p_2 \in \dot{\text{CEXT}}(G, \texttt{Prop}),\\
&\qquad \dot{\text{IEXT}}(G, p_1) = \dot{\text{IEXT}}(G, p_2)^{-1}\}
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
\Gamma_8(G) = &\{(p, \texttt{domain}, c) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}), c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class})\\
&\qquad \forall u, v, \in \text{nd}(G), (u, p, v) \in G \Rightarrow u \in \dot{\text{CEXT}}(G, c)\}\\
&\cup \{(p, \texttt{range}, c) : p \in \dot{\text{CEXT}}(G, \texttt{Prop}), c \in \dot{\text{CEXT}}(G, \texttt{rdfs:Class})\\
&\qquad \forall u, v, \in \text{nd}(G), (u, p, v) \in G \Rightarrow v \in \dot{\text{CEXT}}(G, c)\}
\end{aligned}
\tag{22}
$$

The final step is tricky, because as we add `subPropOf` and `equivProp` triples, the properties that are sub-properties and equivalent properties to these two change. We observe that in the final result `equivProp` will be a sub-property of `subPropOf` but not a super-property: for example `priorVers` will be a sub-property of `type`. Second, we observe that these two properties will have no other super-properties, except for themselves, since (`priorVers`, `priorVers`) is in their property extension, and not in the property extension of any other property. So we can add `subPropOf` triples by the obvious rule, (which relies on an observation about many triples) and the process is monotonic, we will not add a triple that undoes the reasoning that led to the addition of a triple. As we add triples, `subPropOf` and `equivProp` may gain more subproperties. Thus, we define Γ_9 as a closure, and this closure is built around a graph function Ω which is fairly similar in form to (12).

$$\Omega(G) = \begin{array}{l}\{(p, \texttt{subPropOf}, q) : p, q \in \dot{\text{CEXT}}(G, \texttt{Prop}), \dot{\text{IEXT}}(p, G) \subset \dot{\text{IEXT}}(q, G)\} \\ \cup \{(p, \texttt{equivProp}, q) : (p, \texttt{subPropOf}, q), (q, \texttt{subPropOf}, p) \in G\}\end{array} \tag{23}$$

$$S_G = \left\{ G' : \begin{array}{l} G' \subset (\text{nd}(G))^3, G \cup \Omega(G') \subset G', \\ \forall (s, p, o) \in G' \setminus G, p \in \{\texttt{subPropOf}, \texttt{equivProp}\} \end{array} \right\} \tag{24}$$

$$\Gamma_9(G) = \bigcap S_G \tag{25}$$

The following lemma shows that the above definition does what is required.

Lemma 3. $\Omega(H_9) \subset H_9$.

Proof. Consider $\dot{\Omega}$ given by:

$$\dot{\Omega}(G) = G \cup \Omega(G) \tag{26}$$

Then the sequence $G, \dot{\Omega}(G), \dot{\Omega}^2(G), \ldots$ is monotone increasing and we can consider:

$$\dot{G} = \bigcup_{n=1}^{\infty} \dot{\Omega}^n(G) \tag{27}$$

and $\dot{G} \in S_G$. We also have,

$$\forall G, \forall p \in \text{nd}(G) \setminus \{\texttt{equivProp}, \texttt{subPropOf}\}, \forall G' \in S_G, \\ \dot{\text{IEXT}}(G, p) = \dot{\text{IEXT}}(\Gamma_9(G), p) = \dot{\text{IEXT}}(G', p) \tag{28}$$

Take any $G' \in S_{H_8}$, we wish to show that there is no $(s, p, o) \in \dot{H}_8 \setminus G'$.

If $(s, \texttt{subPropOf}, o) \in \dot{H}_8 \setminus G'$, then from (28), at least one of s and o must be either `subPropOf` or `equivProp`. They cannot both be, because both $\dot{\Omega}(H_8)$ and G' contain three of the four possibilities, and the last (`equivProp`, `subPropOf`, `subPropOf`), cannot occur in any $\dot{\Omega}^n(H_8)$ because they all contain (`priorVers`, `subPropOf`, b_9), and not (`priorVers`, `equivProp`, b_9). To show that s cannot be `subPropOf` or `equivProp`, we first consider the special case when triples with predicate o have been introduced in the previous steps relating classes to classes, or properties to classes. There is no node n (including the OWL vocabulary or the blank nodes), for which we can rule out that (n, `type`, `rdfs:Class`) $\in H_8$ because there may be an unusual metamodelling triple such as (`type`, `domain`, `rdfs:Class`) or (`Resource`, `subClassOf`, `rdfs:Class`) which would have introduced such a triple in step 2. Thus, we have to carefully consider the cases:

$$s \in \{\texttt{subPropOf}, \texttt{equivProp}\} \tag{29}$$

$$o \in \{\texttt{subClassOf}, \texttt{equivClass}, \texttt{disjointWith}, \texttt{complmntOf}, \texttt{domain}, \texttt{range}\} \tag{30}$$

On a triple by triple basis, we can show:

$$(b_8, b_8) \in \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{subPropOf}) \cap \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{equivProp}) \quad (31)$$

$$(b_8, b_8) \notin \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{domain}) \cup \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{range}) \quad (32)$$

$$(\texttt{priorVers}, \texttt{backComp}) \in \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{subPropOf}) \cap \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{equivProp}) \quad (33)$$

$$(\texttt{priorVers}, \texttt{backComp}) \notin \begin{array}{r} \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{subClassOf}) \cup \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{equivClass}) \\ \cup \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{complmntOf}) \end{array} \quad (34)$$

$$(\texttt{priorVers}, \texttt{priorVers}) \in \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{subPropOf}) \cap \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{equivProp}) \quad (35)$$

$$(\texttt{priorVers}, \texttt{priorVers}) \notin \text{IE}\dot{\text{X}}\text{T}(H_8, \texttt{disjointWith}) \quad (36)$$

which disposes of these cases. For other o we have $(b_8, b_8) \notin \text{IE}\dot{\text{X}}\text{T}(H_8, o)$, so that $s \notin \{\texttt{subPropOf}, \texttt{equivProp}\}$, as desired, so that $o \in \{\texttt{subPropOf}, \texttt{equivProp}\}$.

Noting,

$$\text{IE}\dot{\text{X}}\text{T}(\dot{H}_8, \texttt{equivProp}) \subset \text{IE}\dot{\text{X}}\text{T}(\dot{H}_8, \texttt{subPropOf}) \quad (37)$$

w.l.o.g. we have $o = \texttt{subPropOf}$.

Ex hypothesis, there is some least n with

$$(s, \texttt{subPropOf}, \texttt{subPropOf}) \in \dot{\Omega}^n(H_8) \setminus G' \quad (38)$$

W.l.o.g. s is such that there is no $s' \notin \{\texttt{subPropOf}, \texttt{equivProp}\}$ with $m < n$ and

$$(s', \texttt{subPropOf}, \texttt{subPropOf}) \in \dot{\Omega}^m(H_8) \setminus G' \quad (39)$$

Thus,

$$\text{IE}\dot{\text{X}}\text{T}(\dot{\Omega}^{n-1}(H_8), \texttt{subPropOf}) \subset \text{IE}\dot{\text{X}}\text{T}(G', \texttt{subPropOf}) \quad (40)$$

and we have:

$$\begin{aligned} \text{IE}\dot{\text{X}}\text{T}(G', s) &= \text{IE}\dot{\text{X}}\text{T}(\dot{\Omega}^{n-1}(H_8), s) \\ &\subset \text{IE}\dot{\text{X}}\text{T}(\dot{\Omega}^{n-1}(H_8), \texttt{subPropOf}) \subset \text{IE}\dot{\text{X}}\text{T}(G', \texttt{subPropOf}) \end{aligned} \quad (41)$$

and

$$(s, \texttt{subPropOf}, \texttt{subPropOf}) \in \Omega(G') \subset G' \quad (42)$$

giving a contradition. We then directly have that if $(s, \texttt{equivProp}, o) \in \dot{H}_8$ then $(s, \texttt{equivProp}, o) \in G'$, so that $\dot{H}_8 \subset G'$.

Then, $\Gamma_9(H_8) = \dot{H}_8$, and the lemma follows.

9.4 Phase 2: proof

Proof (of lemma 1). The RDF and RDFS axiomatic triples are all in H_0 by construction, (and hence all of the H_i) since none of the built-in vocabulary was replaced with literals since the original RDF graph G is well-formed.

We consider each of the Γ_j in turn, proving $\forall i, j$, s.t. $1 \leq j \leq i \leq 9$, that $\Gamma_j(H_i) \subset H_i$, remembering that if $i \leq j$ then $H_i \subset H_j$.

Γ_1: Trivial.

Γ_2: This follows by definition for $i = 2$. In the later cases we have to check that any new triples introduced do not introduce new R or R' consequences. None of the new triples introduced in steps 3 to 9 match the premises of rule rdfs6, rdfs10, rdfs12, odr1, odr2, odr3 or odr4. Triples matching rdfs13 are introduced in step 4, along with the consequence of the rule. Every predicate introduced later has already been used as a predicate in Γ_1, so that rdf1 introduces no new consequences. Similarly, rdfs4a and rdfs4b have no new effect, because all the nodes have previously had one of these rules apply (possibly via rdf1 and rdfs6). Thus we can ignore all these rules. We can also ignore rdfs2 and rdfs3, because the later steps only ever introduce a triple (s, p, o) when either p is in the OWL vocabulary, and hence no `range` or `domain` triple for p is present, or when there have already been triples (s', p, o) and (s, p, o') introduced in earlier steps (some in the RDF axiomatics triples, some in our tables of triples, and some in R'). `subPropOf` triples, which may match rdfs7, are introduced in the last step, but only when the consequence of the rule is already present. Similarly, the `subClassOf` triples introduced in step 5, do not introduce new consequences for rule rdfs9. The `subClassOf` triples introduced in step 4 are more problematic. These potentially match rule rdfs9. The conclusion of rdfs9 is required in the definition of Γ_4, for the introduction of the `subClassOf` triple. Any of the triples introduced in steps 3 to 8, may match the wild-card triple in rdfs7, but they all either have a predicate from the OWL vocabulary, which does not have any explicit proper superproperties, or a property which has no explicit proper superproperties by virtue of the definition of well-formed graph.

Γ_3: This follows from $\mathrm{nd}(H_i) = \mathrm{nd}(H_1)$.

Γ_4: In steps 4 through to 9, new triples are introduced only for 14 predicates. The initialization triples are carefully arranged so that the final behaviour of these predicates is known in advance, in terms of whether they are functional, inverse functional, transitive or symmetric. This relies on the initialization table having the private vocabulary, from the OWL namespace, and using local blank nodes, neither of which occur in H_0. Typically, the initialization triples are arranged so that whenever possible each of these predicates in the final graph has as few as possible of the behaviours. For example, `intersectionOf` is typically transitive, since most RDF graphs do not contains nodes that represent both a class and a list. But they can, and we have not prohibited G_0 from having such behaviour. Thus our initialization data does include such a case, and enough (correct) `intersectionOf` triples to demonstrate in step 4 that it is not transitive. A few of these 14 predicates gain necessary behaviours when they are added. For example, `subPropOf` is transitive, in H_9. But it is not in H_3. In these cases, explicit initialization triples are present declaring the right answer in advance, thus ensuring the lemma holds for that predicate. Tables 4 and 5 discusses each

of these 56 cases in detail. Each cell shows either a number 1 or 4 to indicate that the corresponding triple $(p, \texttt{type}, c)$ is first present in H_1 or H_4, or table 4 shows a single triple that is not present in any of the H_i, whereas table 5 shows two triples that are present in all of the H_i for $i \geq 3$, which prevent $\Gamma_4(H_i)$ from containing $(p, \texttt{type}, c)$. That none of the triples in table 4 are introduced in steps 4 to 9, can be shown on a case by case basis, considering the initialization data, along with the observation that none of the blank nodes or the OWL vocabulary occurs in H_0. This is left as a rather tedious exercise for the reader.

p	Symmetric	Transitive
`type`	(`List` `type` b_4)	(`TransProp` `type` b_3)
`subClassOf`	(`rdfs:Class` `subClassOf` `Restrict`)	1
`equivClass`	4	4
`complmntOf`	4	(`Resource` `complmntOf` `Thing`)
`disjointWith`	4	(`Resource` `disjointWith` `Resource`)
`unionOf`	(b_2 `unionOf` `owl:Class`)	(`rdfs:Class` `unionOf` `nil`)
`intersectionOf`	(b_2 `intersectionOf` `owl:Class`)	(`rdfs:Class` `intersectionOf` b_5)
`oneOf`	(`nil` `oneOf` b_2)	(b_6 `oneOf` `nil`)
`distinctMembers`	(b_3 `distinctMembers` b_1)	4
`inverseOf`	1	(b_{12} `inverseOf` b_{12})
`range`	(b_{10} `range` b_9)	(b_9 `range` b_8)
`domain`	(b_{10} `domain` b_9)	(b_{12} `domain` b_8)
`subPropOf`	(b_9 `subPropOf` `priorVers`)	1
`equivProp`	1	4

Table 4. Concerning $\Gamma_4(H_3)$. Triples shown are absent from H_3 and H_9

Γ_5: No new `type` triples are introduced in steps 5 to 9, so there is nothing to prove.

Γ_6: No new relevant triples are introduced in steps 6 to 9, so there is nothing to prove.

Γ_7: All of the predicates introduced in steps 7 to 9 have predicate that have property extension in the initialization tables involving either the OWL vocabulary or blank nodes. Thus their inverse cannot arise from triples in H_0; and for each such predicate p there is at least one of the initialization triples (s, p, o) s.t. there is no predicate q with (o, q, s) in the initialization triples, or their consequences in any of the steps of the construction.

Γ_8: The conditions for `domain` and `range` in Γ_8, are monotonic decreasing, in that, as we add more triples, no new `domain` and `range` triples arise.

p	Functional	Inverse Func.
type	(comment type Resource) (comment type ObjProp)	(seeAlso type ObjProp) (someValuesFrom type ObjProp)
subClassOf	(Restrict subClassOf rdfs:Class) (Restrict subClassOf Restrict)	(Restrict subClassOf rdfs:Class) (Datatype subClassOf rdfs:Class)
equivClass	(Thing equivClass Thing) (Thing equivClass Resource)	(Thing equivClass Thing) (Resource equivClass Thing)
complmntOf	(Nothing complmntOf Thing) (Nothing complmntOf Resource)	(Resource complmntOf Nothing) (Thing complmntOf Nothing)
disjointWith	(Nothing disjointWith Resource) (Nothing disjointWith Nothing)	(Resource disjointWith Nothing) (Nothing disjointWith Nothing)
unionOf	(rdfs:Class unionOf b_3) (rdfs:Class unionOf b_2)	(owl:Class unionOf b_2) (rdfs:Class unionOf b_2)
intersectionOf	(rdfs:Class intersectionOf b_2) (rdfs:Class intersectionOf b_3)	(rdfs:Class intersectionOf b_2) (owl:Class intersectionOf b_2)
oneOf	(b_6 oneOf b_2) (b_6 oneOf b_3)	(b_7 oneOf b_3) (b_6 oneOf b_3)
distinctMembers	(b_1 distinctMembers b_2) (b_1 distinctMembers b_3)	4
inverseOf	(priorVers inverseOf priorVers) (priorVers inverseOf backComp)	(incompat inverseOf backComp) (priorVers inverseOf backComp)
range	(priorVers range Datatype) (priorVers range Nothing)	(maxCardinality range Resource) (equivClass range Resource)
domain	(rest domain Resource) (rest domain List)	(value domain Resource) (oneOf domain Resource)
subPropOf	(_1 subPropOf _1) (_1 subPropOf member)	(priorVers subPropOf b_9) (b_9 subPropOf b_9)
equivProp	(priorVers equivProp priorVers) (priorVers equivProp backComp)	(incompat equivProp backComp) (priorVers equivProp backComp)

Table 5. Concerning $\Gamma_4(H_3)$. Triples shown are present in H_3 and H_9.

Γ_9: This follows from lemma 3.

Proof (of lemma 2). That $M_i(\Gamma_i(H_9)) \subset M_i(H_9)$ and $\Omega(H_9) \subset M_9(H_9)$, follows directly from the previous lemma and lemma 3.

To show $M_i(H_9) \subset M_i(\Gamma_i(H_9))$, we need to show, both $M_i(H_{i-1}) \subset M_i(\Gamma_i(H_9))$, and that $M_i((\Gamma_i(H_{i-1})) \subset M_i(\Gamma_i(H_9))$, since no new matching triples are introduced after step $i = 3, \ldots 8$.

We start with the latter, the proof procedes like the previous proof by considering each of the Γ_i in turn. As in the previous proof, there is nothing to show for Γ_3, Γ_5 and Γ_6.

Γ_4: We defer this step until we have established the other claims of the lemma.

Γ_7: We only need to show that any inverses for `inverseOf`, `domain`, `range`, `subPropOf` present in H_6 have not been removed by the changes in their property extension in H_9. By considering the initialization triples, involving the OWL vocabulary and blank nodes, they have no inverses in H_6, so there is nothing to prove.

Γ_8: In H_7, by considering the initialization triples, any domain of `domain`, `range`, `subPropOf` or `equivProp`, and any range of `subPropOf` or `equivProp`, must be

`Prop` or a superclass thereof. Similarly any range of `domain` or `range` must be `rdfs:Class` or a superclass of `rdfs:Class`. These are also true for the additional triples in H_8 and H_9.

To show, $M_i(H_{i-1}) \subset M_i(\Gamma_i(H_9))$ and also $M_9(H_8) \subset M_9(\Omega(H_9))$, we start by observing that for `subClassOf` and `subPropOf`, we know that all the $H_i, i > 2$ are R-closed. Thus we need only consider triples for the OWL vocabulary added in the initialization tables. Each of the axiomatic triples is true and follows the construction in the relevant Γ_i and so is in $\Gamma_i(H_9)$. Each of the other triples is true and follows the construction in the relevant Γ_i given the other triples in the that table, and noting the lack of unexpected triples for the OWL vocabulary and the blank nodes. In particular, whenever one such triple needed for H_9 is not constructed in the appropriate $\Gamma_i(H_{i-1})$ it is explicitly present in the initialization tables.

Since the case $i = 4$ is the most difficult, we will discuss it in more detail, along with the deferred question of whether $M_4((\Gamma_4(H_3)) \subset M_4(\Gamma_4(H_9))$. These are both shown as the positive cases in tables 4 and 5.

Given that we have already shown all the other claims of this lemma, almost all of these positive cases correspond to necessary conditions on H_9. The only contingent cases are for `distinctMembers`, which is shown as both transitive and inverse functional. This follows in H_9 because b_1 is the unique member of $\text{CE}\dot{\text{X}}\text{T}(H_9, \texttt{AllDifferent})$ and b_1 is not in $\text{CE}\dot{\text{X}}\text{T}(H_9, \texttt{List})$.

As a final syntactic lemma before we proceed to the semantics:

Lemma 4. $\text{CE}\dot{\text{X}}\text{T}(H_9, \texttt{Literal}) \subset L_{\text{plain}} \cup \mathcal{L}_D$*, i.e. there are no blank nodes or IRI nodes in the class extension of* `Literal`*.*

Proof. This is true by construction of H_0. The only steps in the construction of H_9 that might possibly introduce such triples is 2, when we take an $R \cup R'$-closure. However, H_0 is R-closed (because G_3 was), and so the only relevant new triples that might be at the beginning of new derivations of a triple $(x, \texttt{type}, \texttt{Literal})$ must be one of the initialization triples, or a consequence of a rule in R'. All these triples have predicate being one of the protected vocabulary, or `first` or `rest` or a blank node or in the OWL vocabulary. None of these have declared range or domain `Literal`. Also by a similar argument none of the classes used in the new triples have declared superclass `Literal`, since if they did one of `type`, `subPropOf` or `subClassOf` would have declared `domain` or `range` of `Literal`, via the RDF and RDFS axiomatic triples.

10 An OWL Full$_c$ interpretation of H_9 and G_0

We define a simple interpretation $\mathcal{J}$ of $V_{\mathcal{J}} = \text{vocab}(H_9) \cup \text{vocab}(G_3)$, using the datatype map D, the D-interpretation $\mathcal{I}$ used in section 9.2, via the auxiliary node mapping function ψ used in that section.

We take the universe U as the nodes of H_9 except those that represent literals or datatypes, and we define two auxiliary functions: χ, a one-one mapping between $\mathrm{nd}(H_9)$ and U; and θ, that extends the domain of χ to include $\mathrm{nd}(G_3)$

$$U = (\mathrm{nd}(H_9) \setminus (\mathrm{dom}(D) \cup \mathcal{L}_D)) \cup \mathcal{V}_D \cup \mathrm{ran}(D) \tag{43}$$

$$\chi : \mathrm{nd}(H_9) \to U \tag{44}$$

$$\chi(x) = \begin{cases} d & (x,d) \in D \\ \mathcal{LV}(d) & x \in \mathcal{L}_D \\ x & \text{otherwise} \end{cases} \tag{45}$$

$$\theta : \mathrm{nd}(H_9) \cup \mathrm{nd}(G_3) \to U \tag{46}$$

$$\theta(x) = \begin{cases} \chi(x) & x \in \mathrm{nd}(H_9) \\ \chi(\psi(x)) & x \in \mathrm{nd}(G_3) \end{cases} \tag{47}$$

(47) is well-defined, because if $x \in \mathrm{nd}(H_9 \cap \mathrm{nd}(G_3))$ then $x = \psi(x)$.

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

$$\mathrm{IR}_{\mathcal{J}} = U \qquad \mathrm{IS}_{\mathcal{J}}(x) = \theta(x) \qquad \mathrm{IL}_{\mathcal{J}}(x) = \theta(x) \tag{48}$$

$$\mathrm{IP}_{\mathcal{J}} = \{\chi(p) : p \in \dot{\mathrm{CEXT}}(H_9, \texttt{Prop})\} \tag{49}$$

$$\mathrm{IEXT}_{\mathcal{J}}(x) = \{(\chi(s), \chi(o)) : (s, \chi^{-1}(x), o) \in H_9\} \tag{50}$$

$$\mathrm{LV}_{\mathcal{J}} = \{\chi(l) : l \in \dot{\mathrm{CEXT}}(H_9, \texttt{Literal})\} \tag{51}$$

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_9 and G_3.*

Proof. In both cases we use θ to interpret blank nodes, and we note that θ is the identity on plain literals, which are all in $\mathcal{L}_D$.

Thus, for $(s,p,o) \in H_9$, we have:

$$\begin{aligned} (\mathcal{J}+\theta)(s,p,o) &= (\theta(s), \theta(p), \theta(o)) \\ &= (\chi(s), \chi(p), \chi(o)) \\ &= \text{true} \end{aligned} \tag{52}$$

And, for $(s,p,o) \in G_3$, we have:

$$\begin{aligned} (\mathcal{J}+\theta)(s,p,o) &= (\theta(s), \theta(p), \theta(o)) \\ &= (\chi(\psi(s)), \chi(\psi(p)), \chi(\psi(o))) \\ &= \text{true} \qquad \text{since } (\psi(s), \psi(p), \psi(o)) \in H_0 \subset H_9 \end{aligned} \tag{53}$$

To show that this is an OWL Full$_c$ interpretation we start with the "if and only if" conditions on the four RDFS vocabulary items, and then return to show that $\mathcal{J}$ is an RDFS and D interpretation, before continuing with the other conditions on OWL Full$_c$ interpretations.

Lemma 6. *$\mathcal{J}$ satisfies the "If-and-only-if conditions for* `subClassOf`, `subPropOf`, `domain` *and* `range`*" from [1]*

Proof. We ignore the side conditions on x and y in the statement of these conditions, and prove the stronger result without any. Since χ is a one-one mapping from U to $\mathrm{nd}(H_9)$ these follow directly from lemma 2, with $i = 5, 8, 9$.

10.1 $\mathcal{J}$ is a D-interpretation

We are going to defer to ter Horst's proof that $\mathcal{J}$ is a D* interpretation, and then show the additional constraints of a D-interpretation.

To do that, we have to show that H_9 is closed under ter Horst's rules, which follow Hayes'. However, since we have simplified the framework by using generalized graphs a literal node uses another literal (often itself) as its surrogate, instead of the blank node used by ter Horst. The surrogate for a literal l in our framework, is $\mathcal{VL}_D(\mathcal{LV}_D(l))$. For literals in $\mathrm{nd}(H_9)$ this is the identity.

Thus H_9 is closed under gl and lg. It is also closed under rdfs1 and rdf2-D because $\psi(G_D) \subset H_9$. It is closed under rdfs2, rdfs3, rdfs5, rdfs6, rdfs7, rdfs8, rdfs9, rdfs10, rdfs11 by the previous lemma. It is closed under the rules in R because $\Gamma_2(H_9) \subset H_9$.

Thus we can apply ter Horst's technique, to take the D* Herbrand closure of H_9, which is H_9, since it is closed under his rule set. Thus we find in his lemma 4.10, a D* interpretation $S(H_9)$ of H_9 (note H_9 does not contain a D-clash, otherwise known as an ill-typed literal). His $S(H_9)$ differs from $\mathcal{J}$ only in that $\mathrm{IS}_{\mathcal{J}}$ is defined on more IRIs and $\mathrm{IL}_{\mathcal{J}}$ is defined on more literals (those from $\mathrm{vocab}(G_3)$). These differences do not impact the semantic conditions, so we have:

Lemma 7. *$\mathcal{J}$ is a D* interpretation.*

Lemma 8. *$\mathcal{J}$ is a D interpretation.*

Proof. We have to prove the following additional constraint from [4]: that $\forall (a, d) \in D$ we have $\mathrm{CEXT}_{\mathcal{J}}(d) = \mathcal{V}_d \subset \mathrm{LV}_{\mathcal{J}}$.

We observe that $\mathcal{V}_d \subset \mathcal{V}_D \subset \mathrm{LV}_{\mathcal{J}}$.

For every $v \in \mathcal{V}_d$, the triple $(\mathcal{VL}_d(v), \texttt{type}, a)$ is in G_D, G_3, H_0 and H_9, so that $\mathcal{V}_d \subset \mathrm{CEXT}_{\mathcal{J}}(d)$.

Suppose that $v \in \mathrm{CEXT}_{\mathcal{J}}(d) \setminus \mathcal{V}_d$. Then $v \in \mathrm{CEXT}_{\mathcal{J}}(\texttt{Literal})$ (since $(a, \texttt{subClassOf}, \texttt{Literal})$ is in H_9). Thus by lemma 4, $\chi^{-1}(v)$ is a literal node. But no new literal nodes were introduced during the construction, and no new `type` triples were added to literal nodes (except possible `FunctionalProp` etc.) Thus this triple was present in H_0, giving a contradiction.

10.2 $\mathcal{J}$ is an OWL Full$_c$ interpretation

Lemma 9. *$\mathcal{J}$ is an OWL Full$_c$ interpretation.*

Proof. We need to demonstrate that $\mathcal{J}$ satisfies the conditions from [1], sections 5.2 and 5.3, along with the additional modifications made in our section 3.

We start with section 5.3, and observe that the class extensions of `Thing`, `ObjProp`, and `owl:Class` are determined precisely by the axiomatic triples:

```
rdfs:Class   subClassOf   owl:Class
Prop   subClassOf   ObjProp
Resource   subClassOf   Thing
```

which gives the three conditions of that section.

We now work through the many conditions in section 5.2 in document order.

The non trivial and/or definitional parts of the first table "Conditions concerning the parts of the OWL universe and syntactic categories" are satisfied (in order) because of the following axiomatic triples, in conjunction with the conditions of section 5.3.

```
Datatype subClassOf rdfs:Class
Restrict subClassOf rdfs:Class
Nothing   type   rdfs:Class
Literal type Datatype
Prop   subClassOf   ObjProp
DataProp   subClassOf   Prop
AnnProp   subClassOf   Prop
OntProp   subClassOf   Prop
Ontology type rdfs:Class
b_1   type   AllDifferent
nil type List  (RDF axiom)
```

In addition, we note that $\mathrm{LV}_{\mathcal{J}} = \textsc{cext}_{\mathcal{J}}(\texttt{Literal})$, and that no triple $(s, \texttt{type}, \texttt{Nothing}) \in H_9$.

The built-in classes and properties are all in $\textsc{cext}_{\mathcal{J}}(\texttt{rdfs:Class})$ and $\textsc{cext}_{\mathcal{J}}(\texttt{Prop})$ because they all appear appropriately in the axiomatic or other triples, and the rules of construction for H_9 (e.g. the domain and range axioms on `type` and `subClassOf`) ensure the result.

The built-in annotation and ontology properties are explicitly listed in the axioms.

Concerning the "Characteristics of OWL classes, datatypes, and properties":

- The conditions on `owl:Class`, `ObjProp`, and `AnnProp` are trivial since $\textsc{iot} = R_{\mathcal{J}}$.
- The condition on `Datatype` is satisfied by virtue of H_9 being closed under rule rdfs13.
- The condition on `DataProp` is satisfied since the class is empty.
- The condition on `OntProp` is satisfied since every member of the class has empty property extension.
- The conditions on `FunctionalProp`, `InvFunProp`, `SymProp` and `TransProp` are satisfied by virtue of lemma 2.

This same lemma ensures that both the "characteristics of OWL vocabulary relating to equivalence" and the "Conditions on OWL vocabulary related to boolean combinations and sets" are satisfied.

The second of the further conditions on owl:oneOf is met trivially, since we only add `oneOf` triples whose subject is a class. The first is ensured by making all non-empty (see section 3.1) finite classes of literals of `type Datatype`, in Γ_4, and lemma 1.

The "Conditions on OWL restrictions" are met trivially, since all of the properties have empty property extension.

The comprehension principles are excluded from OWL Full_c.

The requirements from section 3 are met because: $b_1 \in \text{CEXT}_{\mathcal{J}}(\texttt{AllDifferent})$, and the condition on `distinctMembers` corresponds to the constraint in Γ_6.

That's it.

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*, as in table 6. Each of these

For any x, y from nd(G) *or* $\mathcal{L}_D$ *or* dom(D) *or the OWL, RDF or RDFS vocabularies, add a new blank node b_i and:*		
1: (b_1 `rest` x)	(b_1 `first` y)	
2: (b_2 `type owl:Class`)	(b_2 `unionOf` x)	
3: (b_3 `type owl:Class`)	(b_3 `intersectionOf` x)	
4a: (b_4 `type owl:Class`)	(b_4 `oneOf` x)	
4b: (b'_4 `type DataRange`)	(b'_4 `oneOf` x)	
5: (b_5 `type owl:Class`)	(b_5 `complmntOf` x)	
6: (b_6 `type Restrict`)	(b_6 `onProperty` x)	(b_6 `allValuesFrom` y)
7: (b_7 `type Restrict`)	(b_7 `onProperty` x)	(b_7 `someValuesFrom` y)
8: (b_8 `type Restrict`)	(b_8 `onProperty` x)	(b_8 `hasValue` y)
9: (b_9 `type Restrict`)	(b_9 `onProperty` x)	(b_9 `minCardinality` y)
10: (b_{10} `type Restrict`)	(b_{10} `onProperty` x)	(b_{10} `maxCardinality` y)
11: (b_{11} `type Restrict`)	(b_{11} `onProperty` x)	(b_{11} `cardinality` y)

Table 6. Comprehension rules for the DL syntactic subset

is such that, given nodes that exist in a graph (or in the vocabularies being used) it produces 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 these rules are such 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 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 [14] 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. A specification that does not approximate to the current actual practice, and is not expected to influence future practice, is not a specification at all.

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.

13 Conclusions

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. Technical Report, HP Labs (2008) HPL-2008-58.
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. Malhotra, A., Biron, P.V.: XML Schema Part 2: Datatypes Second Edition. W3C Recommendation, W3C (October 2004) http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.
12. Carroll, J.J., Pan, J.Z.: XML Schema Datatypes in RDF and OWL. W3C WG Note, W3C (2006) http://www.w3.org/TR/2006/NOTE-swbp-xsch-datatypes-20060314/.
13. Carroll, J.J.: XML Schema Datatypes in RDF (2002) http://lists.w3.org/Archives/Public/www-archive/2002Nov/att-0092/.
14. 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)	(backComp 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 7. 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 backComp)	(rdfs:Class unionOf b_2)	(b_8 b_8 b_3)
(incompat inverseOf backComp)	(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 backComp)	(rdfs:Class intersectionOf b_2)	(b_4 b_{10} b_5)
(incompat equivProp backComp)	(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)	(b_9 type priorVers)
(rdfs:Class type b_3)	(b_9 inverseOf b_{12})	(priorVers subPropOf backComp)

Table 8. Additional triples assumed in the proof.