Linked data schemata: fixing unsound foundations

Kevin Chekov Feeney, Rob Brennan and Gavin Mendel Gleason
Knowledge and Data Engineering Group & ADAPT Centre, School of Computer Science & Statistics, Trinity College Dublin, Ireland

Abstract. This paper describes our tools and method for an evaluation of the practical and logical implications of combining common linked data vocabularies into a single local logical model for the purpose of reasoning or performing quality evaluations. These vocabularies need to be unified to form a combined model because they reference or reuse terms from other linked data vocabularies and thus the definitions of those terms must be imported. We found that strong interdependencies between vocabularies are common and that a significant number of logical and practical problems make this model unification inconsistent. In addition to identifying problems, this paper suggests a set of recommendations for linked data ontology design best practice. Finally we make some suggestions for improving OWL's support for distributed authoring and ontology reuse.

Keywords: Linked Data, Reasoning, Data Quality

1. Introduction

One of the central tenets of the linked data movement is the reuse of terms from existing well-known vocabularies [1] when developing new schemata or datasets. The Semantic Web infrastructure and the RDF, RDFS and OWL languages support this with their inherently distributed and modular nature. Linked data schemata which reuse vocabularies constitute a knowledge model based on multiple, independently devised ontologies that often exhibit varying definitional semantics [2]. In order to reason about linked data datasets – for example to validate that a dataset correctly uses a term from another vocabulary, a basic requirement is the ability to create a unified knowledge model which combines the referenced ontologies and vocabularies. For example, the Asset Description Metadata Schema (adms) ontology contains the triple:

```
adms:Asset  rdfs:subClassOf  dcat:Dataset
```

In order to validate any dataset which uses the adms:Asset term we must combine the adms ontology and the dcat ontology in order to ensure that dcat:Dataset is a valid class.

There are, however, significant theoretical and practical problems in creating a sound and consistent logical model from the vocabularies typically used in linked data. For example, linked data often freely mixes references to ontologies defined in OWL and vocabularies defined in RDFS. As OWL is a syntactic but not semantic extension of RDF, there are well-known problems in creating any unification between RDF models and OWL models [3]. Beyond the theoretical problems, there are significant practical problems in any system where components are developed independently from one another and later combined [4]. For example the well-known ontology hijacking problem (defined by Hogan et al as the redefinition of external classes/properties in a local ontology) is often caused by misuse of OWL's equivalence statements [5].

Although such problems are well known in theory, there has been little work in systematically assessing their practical manifestations in published linked data. This is largely a consequence of the lack of tools which can help to identify the problems, especially given the permissiveness of the OWL open world semantics applied in standard reasoners.

In this paper we investigate the following research questions: (1) to what extent can current heterogeneous linked data vocabularies be unified into consistent logical models that can detect logical or syntactic er-

rors in the resultant schemata? (2) What is the distribution of logical or syntactical schemata errors present in the current Web of Data?

To address such questions we have constructed a reasoner as part of the *Dacura Quality Service*[1], which is designed to consume OWL and RDF linked data schemata and identify potential problems in their specifications. This reasoner uses a much less permissive interpretation than that of standard OWL to find issues which are likely to stem from specification errors, even in cases where they produce valid OWL models. This tool is integrated into a general purpose ontology analysis framework in the Dacura platform [6] which identifies structural dependencies between ontologies and highlights instances of ontology hijacking.

The contribution of this paper is an identification of the challenges present when combining the models of linked data schemata observed in the current Web of Data for validation, a description of the Dacura Quality Service approach to model combination, an extensive quality evaluation of linked data vocabularies in use for logical and syntactical errors and finally a set of recommendations on best practice for constructing linked data vocabularies that will produce unified logical models without errors in the distributed authoring environment of the web.

The structure of the rest of this paper is as follows: in Section 2 we discuss the challenges for linked data schema validation, in Section 3 we discuss related work, in Section 4 there is a description of the approach and validation capabilities of the Dacura Quality Service, Section 5 describes the methodology used for a wide-scale validation of linked data vocabularies conducted with the Dacura Quality Service, then the results of this evaluation are presented in Section 6. In Section 7 we present a set of recommendations for best practice in linked data vocabulary design and specification, and finally Section 8 describes our conclusions and discusses future work.

2. Challenges for Linked Data Schemata Validation

We define a *linked data schema* as the formal description of the structure of a linked data dataset, expressed in RDF, RDFS and/or OWL vocabularies or ontologies, which is sufficiently complete that all individuals in the dataset are described in terms of a consistent logical model of their classes, properties or datatypes. Thus, there are no unspecified terms used in the schema and it is possible to combine all the definitions into a single logical model that respects the specification semantics of the component vocabularies without resorting to an empty schema as a valid model. The schema must be coherent (i.e., have no necessarily unsatisfiable classes), consistent when combined with the data at hand, and mention each of the classes, properties and datatypes present in the data.

According to ISO 9001, validation is the confirmation, through objective evidence, that requirements for a specific intended use or application have been fulfilled [7]. This highlights the central role of evidence, assessment and intended use. The Dictionary of Computing [8] defines data validation as "the process of checking that data conforms to specification". A linked data schema thus enables validation: all terms used must be defined, the definitions must not lead to inconsistency and for some use cases the definitions form the basis for integrity constraints on data described by the schema. In the Dacura approach, validation is the act of rejecting schemata that have no possible models along with the provision of evidence in the form of witness statements that identify the terms that prevent model formation (see Section 4 for details). The purpose of validation is to identify syntactic or logical errors that are often unintended consequences of the ontology engineering process.

2.1. Challenge 1: Heterogeneous Use of RDF, RDFS, OWL and others

OWL DL ontologies describe a formal domain model based on description logic. It is a difficult task to produce a logical model which accurately and correctly encapsulates any non-trivial domain [9]. This has probably influenced the relative popularity of RDFS terms in linked data [10]. In the wild, RDF and OWL are mixed very freely [2], [10]. Polleres et al [10] phrase this as their challenge 2 for reasoning over linked data i.e. linked data is not pure "OWL". In fact some common linked data vocabularies make reference to other ontologies which are not compatible with OWL at all, but specified in raw RDF collection types, or worse DAML or even other esoteric languages (see Section 6 for the evidence we have collected). Since

[1] Source code available at https://github.com/GavinMendelGleason/dacura

these ontologies reference each other's terms, full validation cannot proceed without determining whether the referenced ontologies are themselves consistent and complete.

If linked data was limited to the use of RDFS or OWL DL, or perhaps even some extension of OWL DL which could encompass elements of OWL Full (such as predication over classes and properties) then consistent model checking would be possible. However the problematic historical unification of RDF and OWL as OWL Full has led to an interpretative fissure between it and OWL DL [11]. OWL DL provides a clear model theoretic semantics which allows one to decidably determine whether a given OWL ontology, potentially coupled with instance data, is consistent. By contrast, OWL Full attempts to mix in the very loose syntactic rules of RDF to arrive at a compromise between OWL and RDF and is not decidable due to mixing logical and metalogical symbols. In fact the full unification of RDF and OWL was dropped as a requirement for OWL Full in OWL2 [12].

Two very problematic deficiencies that are encountered when interpreting RDF/RDFS ontologies as OWL for the purpose of model unification are the use of primitive RDF collection types and predication. The primitive RDF properties rdf:first, and rdf:next are seen in the wild but these are used as internal syntactic symbols of OWL. This means that they cannot be used by properties and classes without leading to inconsistency.

The second problem which arises in the wild is the question of predication. In OWL DL, one may not refer to classes of classes, or properties whose domains are themselves classes or properties. This was done in order both to ensure decidability and to avoid well known "Russell-type" paradoxes such as this one derived from [13].

```
ex:noResources a owl:Restriction .
ex:noResources owl:onProperty rdf:type ;
ex:noResources owl:onClass
     ex:hasAResource ;
     ex:noResources
          owl:maxQualifiedCardinality
            "0"^^xsd:nonNegativeInteger ;
ex:hasAResource owl:oneOf
     ( ex:noResources ) .
```

This particular OWL description is satisfied only when it is not, and vice versa. The difficulty arises from the ability to quantify naively over the rdf:type property itself. This is very similar to Russell's use of the set of all sets. There are methods, well known to logicians [14], of allowing predication over classes and predicates by introducing some sort of stratification to the quantification, but no syntax to do so is present in OWL.

In summary the heterogeneity of linked data schemata means that OWL Full is insufficient for validation and the incompatibilities between RDF and OWL DL mean that even if a single model could be constructed, OWL Full would be undecidable and/or incomplete.

2.2. Challenge 2: Linked Data Import and Referencing Semantics

In linked data schemata there are two ways that other ontologies/vocabularies are referenced, either by explicitly including them using owl:imports, or implicitly by making reference to URIs of properties and classes in an external namespace. The meaning of the first is given a precise semantics under OWL DL (which is not unproblematic in its own right as we will see later) and the entire imported ontology is unioned with the current one during reasoning. The second is a widely used convention that URIs are referred to without importation, for example see [15]. This leads to the question of how to validate over such opaque references. This is a serious problem as one could potentially be referring to an instance as a class, or a class as an instance, one could have references to a class which refers to a third ontology which is shared and does not allow sound subsumption, or any number of other such problematic mixing of meanings without any way of checking for correctness.

2.3. Challenge 3: The Impact of Distributed Authoring and Publication

Developing ontologies that can be easily reused in contexts that were not anticipated by the ontology developer is analogous to the software engineering challenge of developing libraries for reuse in situations where they must coexist with a wide variety of other libraries – many of the same principles apply. For example, a basic principle of software engineering is that libraries which use other libraries should not change their behavior for other libraries. Similarly, ontologies which alter other ontologies are dangerous. Gruber expressed one aspect of this as being "able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions" [16]. This sensitivity to ontological hijacking is particularly relevant as OWL's support for

modularity is extremely primitive – the import statement unifies models into a common model that has a global scope.

To understand why ontology hijacking is a problem, consider the following example. Vocabulary A imports vocabulary B and changes the definition of class X within it with the owl:equivalentClass predicate. Vocabulary C also imports ontology B and uses class X, then imports vocabulary A to use an unrelated term within it. Unless the author of C carefully checks the definition of A, they will find themselves unknowingly using a modified version of class X which may render vocabulary C as invalid. This is closely analogous to the situation where a software library modifies the behavior of other libraries – a situation which has been widely recognized as breaking good software engineering practices since the 1970s: software libraries should not have external side effects.

Of course if A, B and C are subsequently unified into a single model, then logical inconsistencies can become apparent. Due to the complexity of OWL, these inconsistencies may only be detectable by reasoner and despite the prevalence of OWL terms in linked data vocabularies it is evident from our findings that many creators of these vocabularies do not perform reasoner-based checks.

Linked data's focus on the reuse of independently developed and maintained ontologies introduces other significant practical problems. Ontologies that reuse other ontologies are vulnerable to these referenced ontologies becoming unavailable over time, or changing in ways that render them incompatible [10]. This highlights the weaknesses in OWL and especially RDFS's ontology or vocabulary lifecycle support and the variety of practices observed makes automated approaches untenable for the open Web of Data where many core vocabularies predate even OWL2's limited versioning metadata. Given the wide diversity of contexts in which they have been developed – and the cost and difficulty in maintaining them – there is a significant risk of ontologies degenerating over time due to changes in the availability or structure of their dependent ontologies.

The OWL API [17] is one approach to addressing this problem – it supports *locality of information* allowing one to only treat assertions made in a specified or local context. However, this depends upon all concerned ontologies using this mechanism correctly and ontologies being well structured.

2.4. Challenge 4: Permissivity of OWL and RDFS

OWL and RDFS are an extremely permissive languages – reasoners will create a valid model wherever possible, inferring many elements automatically [18]. Thus a number of OWL and RDFS descriptions which are formally correct contain human errors not intended by the ontology designer, and yet will produce valid models. For example, the following assertions:

```
ex:name rdfs:domain ex:Bear, ex:Pig;
ex:peter a ex:Man;
ex:peter ex:name "Peter".
```

These will create Peter as an instance of a "Man-BearPig" due to OWL and RDFS allowing inference of class axioms that would produce a valid model. This is counter-intuitive to software engineers who assume a class structure that must be declared in advance. Thus, such specification errors are common in practice yet they are not detected by standard reasoners.

3. Related work and how it differs from our work

There is a wide variety of existing research that is relevant to our work but we categorize it here under three main headings: (1) frameworks and approaches for assessing linked data quality, (2) theoretical studies on the unification of RDF and OWL and (3) reasoning and consuming linked data. Each of these is discussed in turn in the subsections below.

3.1. Frameworks and approaches for assessing linked data quality

The underlying framework for current linked data quality assessment has been defined by Zalveri et al. [23]. In terms of their quality framework our current work addresses mainly intrinsic dimensions of the schema – syntactic validity, semantic accuracy (in terms of misuse of properties) and consistency. However we also address the contextual dimension of understandability by checking for human-readable labelling of properties and classes.

Our current work builds upon the previous version of our Dacura data curation platform [6] by extending the simple rule-based data validation implemented in Apache Jena/Java described in our Workshop on Linked Data Quality 2014 publication [19] with a custom reasoner and ACID (Atomic, Consistent, Isolated, Durable) triple-store for validation and data integrity

enforcement. This new component, the Dacura Quality Service, is built in SWI-Prolog on ClioPatria [20] and is described in the next section. An earlier version of the Dacura Quality Service which covered a much smaller set of OWL features was described in a paper at the 2nd Workshop on Linked Data Quality [19]. That paper has been extended here to also include a discussion of the new Dacura Schema Management service, our experimental validation of linked data schemata in the wild and new recommendations for best practice when constructing new linked data vocabularies.

The RDFUnit methodology for test-driven quality assessment by Kontokostas et al. [21] is a SPARQL-based approach to validating linked data schemata and datasets. RDFUnit is very close to being a union of SPIN and the Stardog[2] ICV approach to validation, which is itself the successor to Pellet ICV [22]. RDFUnit is described by Zaveri et al. [23] as being able to detect intrinsic data quality dimensions for syntactic and semantic accuracy but in common with all SPARQL-based approaches the lack of reasoning ability means that it is difficult to detect consistency problems that may be present. For a specific dataset it is possible to manually generate specific SPARQL-based tests that could detect these errors but the effort required is probably prohibitive and is brittle in the presence of schemata change over time. Similar approaches have been taken with SPARQL and SPIN (SPARQL Inferencing Notation) [24] and the Pellet Integrity Constraint Validator (ICV) [22].

Since 2014, the W3C's Data Shapes Working Group has been working on SHACL (Shapes Constraint Language) to describe structural constraints and validate RDF instance data against those. Instance data validation, (except where the data is part of the schema for example when using owl:oneOf to define a class), is outside the scope of this paper. However it is possible that suitable SHACL constraints could be used to validate RDF graphs describing schemata. As with the basic SHACL-based approach to data validation it is unclear why re-stating a specification in another formalism (SHACL) is a good approach to validation – see also [25] for further remarks on the applicability of description logics to constraints.

Luzzu [26] is a stream-oriented linked data quality assessment framework that focuses on data instance centric measurement of user-defined baskets of quality metrics. Although the metrics are expressed in a domain specific language that is described as extensible it would be necessary for the user to write java code to implement the necessary checks. This is not feasible for most users, even knowledge engineers: the user would have to write an OWL reasoner to detect the logical errors in the unified dependency tree of a linked data schema which Dacura identifies. The framework is potentially of great practical use to users of linked data datasets that wish to assess their quality based on a custom basket of measures, but provides very little assistance when assessing the logical soundness of the schema. One innovation of Luzzu is the specification of a Quality Report Ontology to provide machine-readable quality assessment results. Dacura has a similar ontology defined for reporting reasoning errors, the Reasoning Violations Ontology[3].

Much closer to our Dacura schema validation service is Suárez-Figueroa et al.'s OntOlogy Pitfall Scanner, OOPS! [27]. This is a web-based tool for ontology evaluation. It is aimed at detecting ontology anomalies or "worst practices" as a form of automated ontology evaluation. It is based on evaluation of an input ontology against a catalogue of common errors or pitfalls seen in ontologies. There is very little overlap between the 41 pitfalls currently detected by OOPS! and the set of errors detected by Dacura due to the lack of OWL reasoning or model combination in OOPS. There are some easy to detect "best practice" pitfalls such as "P08 Missing annotations" which Dacura does not currently detect but which are simple extensions of our current best practice rules and will be added in future work. It is interesting to note that in their extensive analysis of current ontologies and the ontology engineering community, OOPS decided, like us, to define class cycles as a potential source of errors in linked data schemata, despite being legal in many cases in OWL 2. The implementation of OOPS! as a restful web service is very attractive and useful for integration into both a basic public webpage for checking and as a service called by other ontology engineering tools.

3.2. Theoretical studies on the unification of RDF and OWL

The challenges for reasoning caused by the unification of RDF and OWL has been extensively discussed

[2] Stardog, http://stardog.com/

[3] http://www.essepuntato.it/lode/owlapi/https://w3id.org/rvo

in the literature, for example see Patel-Schneider and Fensel [28] where, even before OWL was standardized, these incompatibilities were summarized as "defining the model theory of OWL as an extension of the model theory of RDF and representing OWL constructs syntactically in RDF leads to paradoxical situations, i.e., ill-defined model theories for OWL". The authors' five approaches to layering OWL over RDF, including identifying the presence of Russell-type paradox if OWL is directly layered over RDFS as a same-syntax solution. Nonetheless this was the approach adopted in OWL-Full. The ramifications of these decisions live on today in our challenge 1 for validating linked data schemata.

Later, when the OWL standard was agreed, the principal authors of the OWL Semantics documented the "difficult trade-offs" that they made during the design of OWL version 1 [29]. Horrocks et al. identify four broad classes of problems: syntactic, semantic, expressive power and computational problems. While the latter two categories seem less relevant for our current work it was in fact the computational overheads introduced by allowing the use of classes as instances that led to the exclusion of this feature from OWL-DL and as will be seen in our experimental work, under the title of impredicativity, this causes many observed issues in linked data. In Section 6.2 of that paper the solution proposed for dealing with malformed OWL syntax expressed in RDF leads to the creation of additional anonymous classes despite them being "almost certainly not what was intended by the user". This is an example of our challenge 4 (permissivity of OWL) that leads to standard reasoners being unable to detect these issues in linked data schemata without human inspection of the resultant reasoned ontology. In contrast our approach highlights these potential errors and presents them to the user or validator for verification. Finally in the discussion on future extensions despite the admission that the import of ontologies by other is likely to be the norm in the Semantic Web (as we now see in linked data) the OWL import facility is described as "very trivial" and this underpins our challenges 2 and 3 for linked data schemata validation. Both the closed world and unique name assumptions are identified as being desirable in some situations despite being outside the scope of the general OWL model. Our approach to schemata validation and the experimental evidence we have collected demonstrate the practical applicability of these assumptions for validation, even on the open web.

When the OWL standard was revised as OWL2 in 2008, the process was again documented by some of the principal authors [30]. Grau et al. identified additional issues with OWL1 ontology specifications, including an additional complication in the import semantics whereby two (or more) ontologies written in the same OWL species can interact in unpredictable and unintuitive ways when one ontology imports the other, leading to a new ontology that is contained in a different species. OWL 2 introduces the idea of declaration consistency which means all types must be declared, although not syntactically necessary, this forms the basis for additional validation checks to catch mistyping of entity terms. However this approach is not applicable to linked data that is not written in OWL 2. The validation checks performed by our approach can detect such trivial typing errors in general linked data schemata e.g. misspelt names of classes. OWL 2 also improves support for imports by tightening the specification of ontology name URIs as both name and published location of the ontology on the web. However, as our experimental results show (section 6) not all commonly imported ontologies in the current Web of Data are at their stated locations. Ontology versioning management support is also added, but this is still primitive and the species or profile impacts of imports is still unintuitive and unpredictable (challenge 2).

Most recently, as the success of the open linked data movement has become apparent, with billions of triples published, the question of data quality and hence validation has come to the fore [31], [32]. Patel-Schneider [25] discusses the issues and approaches to applying description logic to validation, attacking the claim that closed world or unique name interpretations have no place in the description logic world (and hence within OWL/RDFS) and can be applied to linked data for validation. Although Patel-Schneider focuses on RDFS and a SPARQL-based approach to validation of linked data, our approach adopts some of the same assumptions about closed worlds and unique names in our custom reasoner.

3.3. Reasoning and consuming linked data

The original Semantic Web vision included the goal of applying reasoning at a web scale [33]. Linked data provides the basis of the current Web of Data and so reasoning over it has naturally been tackled by several researchers, see for example [34], [35] and [10]. Given the divergence of linked data from the Semantic Web ideal, a wide variety of non-standard reasoning approaches have been applied from probabilistic techniques [35] to rule-based approaches [34]. Given our

interest in RDFS and OWL-based schemata validation we focus here on approaches that support the RDFS and OWL standards. The challenges for applying reasoning to linked data as laid out by Polleres et al. [10] may be summarized as data scale, data heterogeneity (mixing of OWL DL, OWL Full and RDFS), data inconsistency, data dynamics and extending inference beyond RDFS and OWL. Data scale is less of an issue for our work since we focus on schemata and thus primarily TBox assertions, although some linked data schemata (e.g OpenCyc – see section 6) include instance data which render them very large. Nonetheless the focus of open web reasoning work on operating on billions of triples largely addresses challenges which are out of scope for our validator. Tackling heterogeneity is also a focus of our work (challenge 1) but whereas we aim to identify inconsistent or incomplete schemata in order to fix them and improve their quality, the approach of the reasoning over linked data community is to try and do the best possible with the triples available. While appropriate for their use case, it often leads to strategies that weaken consistency or soundness constraints to make the problem tractable, silently discard problematic triples or conservatively reduce the materialisation of inferred triples compared to completely applying the OWL Direct Semantics [10]. Although there are points of similarity, in general these approaches would produce weaker validation results than our approach since they are not sound and less complete.

4. Linked Data Schemata Validation in the Dacura Quality Service

To meet the challenges of linked data schemata validation, we have developed the Dacura Quality Service (DQS) and the Dacura Schema Manager. Both are integrated into our Dacura platform for data curation described elsewhere [6]. The Dacura Schema Manager (fig. 1) acts as the user interface for loading new linked data schemata into the system. It recursively loads all the implicitly or explicitly imported vocabularies or ontologies from web, creates the master schema based on the union of all referenced terms, gathers statistics and performs some basic quality checks. The validation view of the Dacura Schema Manager allows a user to select specific reasoner-based validation checks, call the DQS through its API to perform the checks and renders the results in human-readable form.

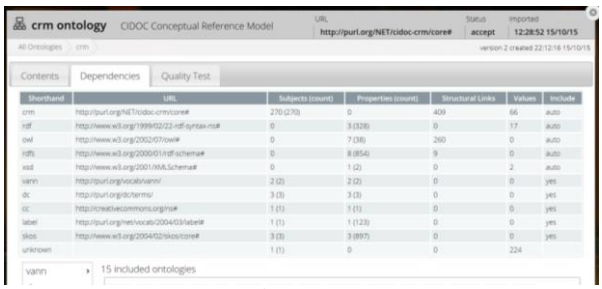

Figure 1: Screenshot of the Dacura Schema Manager

The DQS as an ACID triplestore for the storage of OWL ontologies and instance data. We treat consistency as internal consistency of the OWL ontology as well as consistency of instance data with respect to this ontology. In this way we have produced a triplestore in which stored information always respects the ontology as it is impossible to perform updates which are not consistent. If the schema changes, the instance data must also change in a fashion conformant to the new schema. The DQS is built in SWI Prolog in the ClioPatria Semantic Web infrastructure [36]. The source code is available online under a GPL2 license.

To do this, we have built a custom reasoner as part of the DQS which treats all ontologies which are used as a relatively large but custom fragment of OWL DL (see table 1 for the OWL 2 features implemented so far) subject to additional constraints that increase the ability of the reasoner to deal with the unification of OWL and RDF/RDFS in linked data schemata (challenge 1, challenge 2), detect likely validation errors (challenge 3, challenge 4) and improve efficiency. This fragment of OWL DL has also been shaped by the modelling requirements of ontology development for the Seshat:Global History Databank [37] which is our initial use case, as well as the OWL 2 vocabularies we found most often used in linked data schemata on the Web of Data. The range of support for OWL 2 constructs is substantially increased from our earlier paper [19] which focused on RDFS. It is anticipated that we will continue to extend the support for further OWL 2 features in future work.

The overall strategy of the DQS reasoner is not to prove that there is a possible model for any given ontology but instead to reject ontologies that cannot have a possible model or which are incompletely specified without inferring new classes (as these are often caused by user errors) under a closed world assumption. Due to the ontology import actions of the Dacura Schema Manager, the closed world in this case corresponds to the whole of the Web of Data, at the level of schema specification. We do not claim that the reasoner is sound or complete under OWL DL, just that

it is capable of detecting many errors in linked data schemata, including errors undetectable by standard reasoning. Our approach is supported by building a subsumption prover in SWI Prolog. Due to the complexity of performing subsumption computations with equivalences, we have opted in DQS to ignore non definitional equivalence, hence we do support owl:equivalentClass in one direction but not as a symmetric property. This is because OWL does not distinguish between the definitional and judgmental use of this assertion. In practice this allows users to define a class as a formula of other classes but does not allow them to provide an assertion of two classes being equivalent. In the wild we see the first case used extensively and the second only rarely and when it is it is often problematic (see Table 5 Ontology Hijacking) or recommended to be avoided by ontology engineering best practice (this case is listed as pitfall number P02 in the OOPs catalogue of common pitfalls[4]). Hence we term this as partial support by Dacura for owl:equivalentClass in Table 1.

Dacura does not currently support owl:disjointWith assertions but this only limits the range of validation errors that can be detected rather than introducing false positives so it does not give reason to doubt the errors detected. It is also the case that these assertions are much more important when validating instance data than when validating schemata. Typically we wish to ensure that instance data respects disjointness, not schemata – which is the focus of the work presented here.

It should be noted, however, that in both cases, our dependency analysis tool does correctly recognize that both predicates introduce dependencies between ontologies – however the analysis of the validity of the specified relationships is limited to one-directional equivalence.

We also require that there are no cycles in the declared subsumption of classes or predicates. This again does not give us the full power of OWL DL, however it was very rare that we found any actual intended use of cycles in practice.

Table 1 OWL 2 vocabulary features supported by DQS Reasoner

Language Elements	Supported	**Language Elements (cont.)**	Supported	**Axioms and Assertions (cont.)**	Supported
Classes, Datatype and Restriction		owl:hasValue	Y	**Property Expression Axioms**	
owl:Class	Y	owl:SelfRestriction	N	rdfs:subPropertyOf	Y
owl:intersectionOf	Y	**Special classes**		owl:inverseOf	Y
owl:unionOf	Y	owl:Thing	Y	owl:equivalentProperty	N
owl:complementOf	Y	owl:Nothing	Y	owl:property DisjointWith	Y
owl:oneOf	Y	**Properties**	Y	rdfs:domain	Y
rdfs:Datatype	Y	owl:DatatypeProperty	Y	rdfs:range	Y
owl:datatypeComplementOf	N	owl:ObjectProperty	Y	owl:propertyChain	Y
owl:oneOf	Y	**Special properties**		owl:FunctionalProperty	Y
owl:onDatatype	Y	owl:TopDataProperty	Y	owl:InverseFunctionalProperty	N
owl:withRestrictions	Y	owl:BottomDataProperty	Y	owl:ReflexiveProperty	P
owl:Restriction	Y	owl:TopObjectProperty	Y	owl:IrreflexiveProperty	N
owl:onProperty	Y	owl:BottomObjectProperty	Y	owl:SymmetricProperty	P
owl:onClass	Y	**Individuals**		owl:AsymmetricProperty	P
owl:onDataRange	Y	owl:NamedIndividual	N	owl:TransitiveProperty	Y
owl:onProperties	Y	**Axioms and Assertions**		owl:hasKey	Y
owl:cardinality	Y	**Class Expression Axioms**		**Assertions**	
owl:maxCardinality	Y	rdfs:subClassOf	Y	**owl:NegativePropertyAssertion**	N
owl:minCardinality	Y	owl:equivalentClass	P	owl:sourceIndividual	N
owl:minQualifiedCardinality	Y	owl:disjointWith	N	owl:assertionProperty	N
owl:minQualifiedCardinality	Y	owl:disjointUnionOf	Y	owl:targetValue	N
owl:qualifiedCardinality	Y	**Individual Axioms**		owl:targetIndividual	N
owl:allValuesFrom	Y	owl:differentFrom	N	owl:AllDifferent	N
owl:someValuesFrom	Y	owl:sameAs	N	owl:AllDisjointClasses	N
				owl:AllDisjointProperties	N
Y= Yes, P = Partial, N = No				owl:members	N

[4] http://oops.linkeddata.es/catalogue.jsp

DQS provides an interface to a triple store via HTTP using a simple JSON format for updates (both inserts and deletes) of triples, and of both instance and ontology data. The service responds to updates either with a success message stating that the insertion is consistent, or a message describing the precise reason for failure of consistency according to the reasoner. The reasoner ensures that it builds up a witness of failure which demonstrates the counter-example to consistency satisfaction which can then be used by the client to come up with a suitable strategy for dealing with the failure. The results of our evaluation of linked data schemata (see Sections 5 and 6) were compiled by loading ontologies in the Dacura Schema Manager, and then testing them against the Dacura Quality Service, and then looking at the error reports provided. Next we examine the specific solutions implemented in the Dacura Schema Manager and Dacura Quality Service to address the challenges of linked data schemata validation

Namespace	**Term**
rdf	type
rdfs	range, domain, subPropertyOf, subClassOf, member
owl	inverseOf, unionOf, complementOf, datatypeComplementOf, intersectionOf, oneOf, dataRange, disjointWith, imports, allValuesFrom, someValuesFrom, equivalentClass, equivalentProperty, disjointUnionOf, propertyDisjointWith, members, disjointWith, propertyDisjointWith, onProperty, onClass, propertyChainAxiom

Table 2 – OWL/RDF/RDFS terms that create structural dependencies between ontologies

4.1. Overcoming Challenge 1 (Heterogeneity)

As discussed in the background section, the free mixing of RDF, RDFS and OWL triples gives rise to different interpretations. Our approach is to deliberately misinterpret as OWL the RDF/RDFS classes and properties that are normally outside the scope of OWL-DL when there is no immediate conflict in doing so, e.g. a rdfs:class is treated as equivalent to an owl:class. This doesn't present an insurmountable difficulty for reasoning. Similarly rdf:Property is treated at an equivalent level to owl:DatatypeProperty and owl:ObjectProperty and no overlap is allowed between them. All domains and ranges that are asserted are checked to ensure they support subsumption. Misuse of language features and low level RDF syntax with reserved meaning in OWL such as rdf:List is detected as an error.

This approach is applicable in situations where the data is going to be published only for the combined ontology, or used only internally to a system which interprets the instance data as OWL. This is in line with common practice for linked data but presents potential problems for interoperability of the produced linked data since OWL reasoners might deem it inconsistent due to the fact that we still allow a mix RDFS and OWL and hence are not a proper subset of OWL DL. However, as our experimental results will show, this is necessary for dealing with the commonly used vocabularies on the Web of Data today.

4.2. Overcoming Challenge 2 (Imports)

Since there are a range of ways that linked data schemata reference or import each other, it was necessary to define a mechanism to construct the composite ontology defined by a linked data schemata to enable validation under a closed world assumption. For this reason, we have treated all dependencies to external namespaces as implicit owl:imports.

Dependencies between ontologies were defined as either property dependence or structural dependence:

Property dependence: if an ontology A uses a property from another ontology B, then A is considered to have a dependence on B.

Structural dependence: if an ontology A contains a statement which defines its classes or properties in terms of entities in ontology B, then A is considered to have a structural dependence on B. Table 2 shows the specific OWL terms which we consider create structural links between ontologies.

Other references to external URIs in a schema were ignored.

Having defined what we considered to amount to the class of dependencies between ontologies, the Dacura Schema Manager tool implements these rules

to analyse any given ontology and recursively create its dependency tree, fetch the constituent ontologies or vocabularies and create a union between them for checking by the DQS.

4.3. Overcoming Challenge 3 (Distributed Authoring)

The Dacura Schema Manager detects all dependencies between ontologies as described in the last section. This forms the basis for detecting references to missing or unavailable ontologies. Similarly it can detect namespace violations such as ontology hijacking when they occur in input ontologies. The logical consequences of building unified models from many ontologies are detected by the DQS, especially when local work-arounds have been made that render the unified model inconsistent.

4.4. Overcoming Challenge 4 (OWL Permissivity)

By applying the closed world assumption to the full graph imported from the Web of Data that specifies a linked data schema it is possible to detect orphan classes. These are rejected as incompletely specified (similar to the use of declarations in OWL 2 but without the need to augment existing ontologies with these new declarations). In addition, the detection of subsumption failures and cycles in class or property declarations allows us to detect potential misuse of OWL features.

5. Evaluation Methodology

In order to evaluate the interoperability of the various ontologies and vocabularies which are commonly used by linked data documents, it is first necessary to establish which ontologies or vocabularies are the most common, and by what measure(s) in the Web of Data today. In order to do this we rely on the extensive literature that catalogs the development and makeup of the Web of Data and the live reports from the Linked Open Vocabularies (LOV) site[5] [38]. At the time of writing LOV reported hosting 542 vocabularies.

Table 3: Top 20 Vocabulary Popularity as Reported by LOV, March 2016

Vocabulary	# Vocabularies	# Datasets
dc	439	327
dc11	361	178
foaf	325	249
vann	201	19
skos	200	152
cc	87	21
vs	81	11
schema	48	12
prov	38	39
gr	38	20
geo	37	49
event	36	9
time	30	47
bibo	27	43
void	25	77
org	23	7
adms	23	3
dctype	22	13
sioc	21	18
qb	19	9
frbr	19	12
doap	18	23
voaf	15	2
gn	15	14
ssn	14	0

Despite the undoubted utility of LOV it is clear that it services a specific community of users and so we looked for a wider base of evidence. The ranking in terms of vocabulary reuse is also arguable, compared to the proliferation of a vocabulary's terms in data.

Schmachtenberg et al. in 2014 [39] provided a survey of the results of an extensive crawl of the Web of Data (over 8 million resources were visited) based on the vocabularies registered with datahub.io. This study, as a follow-up to a 2011 baseline, showed an increased reliance by linked data publishers on a small set of core vocabularies compared to 2011. In table 5 of that paper they provide the list of the most often encountered vocabularies in terms of the 18 vocabularies that are used by more than 5% of all datasets. Their list is shown in our Table 4.

Table 4: Most Popular Vocabularies in Linked Data in April 2014 (Schmachtenberg et al.)

Vocabulary	%	Vocabulary	%
rdf	98.22	void	13.51
rdfs	72.58	bio	12.32
foaf	69.13	qb	11.24
dc	56.01	rss	9.76

[5] http://lov.okfn.org/dataset/lov/

owl	36.49	odc	8.48
geo	25.05	w3con	7.6
sioc	17.65	doap	6.41
admin	15.48	bibo	6.11
skos	14.11	dcat	5.82

In addition, they report that of the nearly 1000 datasets visited that only 23% used local vocabularies that are not used in any other dataset while nearly all datasets use vocabularies common to multiple datsets. This shows the consolidation of the Web of Data towards fewer vocabularies as in 2011 64.11% of datasets were found to use local vocabularies not used elsewhere.

Finally in 2011 Hogan et al. surveyed the state of the Web of Data with a crawl of approximately 4 million RDF/XML documents and 1 billion quads [40]. Their Table 2, provided here in abbreviated form as Table 5, shows the top 25 most popular vocabularies, based on the number of instances of each namespace within their analysis dataset.

Table 5: Most Frequently Occurring Vocabularies in Linked Data 2011 (Hogan et al.)

Vocab	Instances	Vocab	Instances
foaf	615,110,022	dc11	6,400,202
rdfs	219,205,911	b2rns	5,839,771
rdf	213,652,227	sioc	5,411,725
b2r	43,182,736	vote	4,057,450
lldpub-med	27,944,794	gn	3,985,276
lldegene	22,228,436	skipinions	3,466,560
skos	19,870,999	dbo	3,299,442
fb	17,500,405	uniprot	2,964,084
owl	13,140,895	eatoc	2,630,198
opium-field	11,594,699	lldlifeskim	2,603,123
mo	11,322,417	ptime	2,519,543
dc	9,238,140	dbpedia	2,371,396
estoc	9,175,574		

Note that both entry 25 (dbp) and 20 (dbo) are DBpedia vocabularies.

In summary the most common vocabularies that appear in all three surveys are: *foaf, dc, sioc* and *skos*; in addition *dc, bibo, qb, doap, geo, void* and *gn, rdf, rdfs* and *owl* appear twice. Hence we must have coverage of all of these core vocabularies to evaluate the foundations of linked data.

From these studies of vocabulary usage, we identified the top 50 most commonly used vocabularies and ontologies in use. However, in order to validate these ontologies, we also need to include all of their dependencies.

5.1. Identifying Dependencies

We applied the Dacura Schema Manager dependencies tool to all of the top 50 ontologies identified. The output of this tool (fig 1) was used to identify the set of ontologies and vocabularies that each ontology depends on directly, then we included these ontologies, identifying the set of ontologies needed by these included ontologies, including them and continuing until all of the dependencies were included or were deemed to be impossible to include. This produced a breadth-first dependency tree for each ontology. This increased the number of ontologies in our analysis set to 91 – shown in Table 6. We then analyzed all these ontologies with the DQS tool to identify to what extent they exhibited problems in terms of creating a unified knowledge model that incorporated them. It should be noted that the ontologies that were included through this dependency analysis are almost all due to the inclusion of the most two most common vocabularies (dc and foaf) and thus most of the dependency tree shown here is common to virtually all linked data vocabularies.

Figure 2 gives an example of the dependency tree for one ontology: Open Annotation [41]. This dependency tree covers 22 of the top 25 vocabularies rated as most popular by LOV in terms of vocabulary reuse (incoming links) as seen in Table 3. This ontology was selected as an example for both practical and theoretical reasons. Practically, we wished to implement a system for the Seshat: Global Hostory Databank in which users could annotate content at a variety of scopes and we wanted to be able to validate instance data which was expressed according to the ontology. Theoretically, it represented a good example of a linked data schema in the wild, as is shown by the analysis above, – it has been constructed by a W3C community group according to the linked data principles, using well known third party vocabularies and ontologies and it is in use in practice. It is ranked by LOV as the 32rd most popular linked data vocabulary overall (from 542 vocabularies) and its dependency tree, as discovered by Dacura, includes 25 of the 31 vocabularies rated as more popular than it by LOV.

Our dependency analysis terminated whenever we came to an ontology that we could not retrieve, either because we discovered that the ontology no longer existed (e.g. WordNet), or because we proved unable to locate a machine-readable version of the ontology on

the internet, after approximately 8 hours of effort in searching. In one case our dependency tree brought us to an ontology that was simply too big for our tools to handle – OpenCyc (rdf/xml file: 246 MB) due to insufficient memory on our test computer. There was only two structural links to this ontology from the rest, so the omission can be considered to be relatively minor. In two cases dependent ontologies were written in DAML, a predecessor of OWL and these ontologies were not automatically analyzed as our tools were not capable of interpreting them. Manual analysis of both revealed that they had no further dependencies.

5.2. Schema Validation

Once the dependency tree of ontologies for each ontology had been established, the composite schema so defined (consisting of the union of all of the imported ontologies) was analyzed by the DQS reasoner and the OOPS! tool for validation errors for each ontology in table 6. See the next section for the results.

Table 6. Ontologies analyzed as part of this work.

shorthand	URL	**Description**
adms	http://www.w3.org/ns/adms#	Asset Description Metadata Schema (ADMS)
ao	http://purl.org/ontology/ao/core#	The Association Ontology
atom	http://bblfish.net/work/atom-owl/2006-06-06/#	Atom syndication format
basic	http://def.see-grid.csiro.au/isotc211/iso19103/2005/basic#	OWL representation of ISO 19103 (Basic types package)
bbc	http://www.bbc.co.uk/ontologies/bbc/	BBC Ontology
bbccor	http://www.bbc.co.uk/ontologies/corecon-cepts	BBC Core Concepts
bbcpro	http://www.bbc.co.uk/ontologies/provenance	BBC Provenance Ontology
bibo	http://purl.org/ontology/bibo/	The Bibliographic Ontology
bio	http://purl.org/vocab/bio/0.1/	BIO: A vocabulary for biographical information
cc	http://creativecommons.org/ns#	Creative Commons
cms	http://www.bbc.co.uk/ontologies/cms/	CMS Ontology
contact	http://www.w3.org/2000/10/swap/pim/con-tact#	Contact: Utility concepts for everyday life
cpa	http://www.ontologydesignpatterns.org/sche-mas/cpannotationschema.owl#	Content Pattern Annotations
crm	http://purl.org/NET/cidoc-crm/core#	CIDOC Conceptual Reference Model
cwork	http://www.bbc.co.uk/ontologies/creative-work	Creative Work Ontology
dbox	http://dublincore.org/documents/dcmi-box/	(Empty) DCMI-Box encoding scheme
dbpedia	http://dbpedia.org/ontology/	The DBpedia Ontology
dc	http://purl.org/dc/terms/	DCMI Metadata Terms – other
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
dcam	http://purl.org/dc/dcam/	Metadata terms related to the DCMI Abstract Model
dcat	http://www.w3.org/ns/dcat#	The data catalog vocabulary
dctype	http://purl.org/dc/dcmitype/	DCMI Type Vocabulary
doap	http://usefulinc.com/ns/doap#	Description of a Project (DOAP) vocabulary
doc	http://www.w3.org/2000/10/swap/pim/doc#	Document vocabulary
dtest	http://www.w3.org/2006/03/test-description#	Test Description Vocabulary
dtype	http://www.linkedmodel.org/schema/dtype#	Specification of simple data types
dul	http://www.loa-cnr.it/ontologies/DUL.owl#	DOLCE+DnS Ultralite
event	http://purl.org/NET/c4dm/event.owl#	The Event ontology
foaf	http://xmlns.com/foaf/0.1/	Friend of a Friend (FOAF) vocabulary
frbr	http://purl.org/vocab/frbr/core#	Expression of Core FRBR Concepts in RDF
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	WGS84 Geo Positioning
geometry	http://data.ordnancesurvey.co.uk/ontol-ogy/geometry/	A ontology to describe abstract geometries.
gn	http://www.geonames.org/ontology#	The Geonames ontology
gr	http://purl.org/goodrelations/v1	Good Relations Ontology

grddl	http://www.w3.org/2003/g/data-view#	GRDDL Gleaning Resource Descriptions
gsp	http://www.opengis.net/ont/geosparql	OGC GeoSPARQL
hcard	http://purl.org/uF/hCard/terms/	HCard Vocabulary
http	http://www.w3.org/2006/http#	A namespace for describing HTTP messages
iana	http://www.iana.org/assignments/relation/	Link Relations
ical	http://www.w3.org/2002/12/cal/ical#	RDF Calendar
icalspec	http://www.w3.org/2002/12/cal/icalSpec#	ICAL specifications
infreal	http://www.ontologydesignpat-terns.org/cp/owl/informationrealization.owl#	Information Realization ontology
irw	http://www.ontologydesignpat-terns.org/ont/web/irw.owl#	The Identity of Resources on the Web ontology
keys	http://purl.org/NET/c4dm/keys.owl#	Musical keys
label	http://purl.org/net/vocab/2004/03/label#	Term definitions for singular and plural label properties
leo	http://linkedevents.org/ontology/	Linking Open Descriptions of Events
log	http://www.w3.org/2000/10/swap/log#	Logic Ontology
mo	http://purl.org/ontology/mo/	The Music Ontology
neogeo	http://geovocab.org/spatial#	A vocabulary for describing topological relations between features
nrl	http://www.semanticdesktop.org/ontologies/2007/08/15/nrl#	NEPOMUK Representational Language
oa	http://www.w3.org/ns/oa#	Open Annotation Data Model
obo	http://purl.obolibrary.org/obo/obi.owl	Ontology for Biomedical Investigations
ont	http://www.w3.org/2006/gen/ont#	An Ontology for Relating Generic and Specific Information Resources
opmv	http://purl.org/net/opmv/ns#	The Core OPMV Vocabulary
org	http://www.w3.org/ns/org#	Core Organization Ontology
ov	http://open.vocab.org/terms/	Open Vocabulary
prv	http://purl.og/net/provenance/ns#	Provenance Vocabulary Core Ontology
prov	http://www.w3.org/ns/prov#	W3C PROVenance Interchange Ontology
qb	http://purl.org/linked-data/cube#	The data cube vocabulary
qudt	http://qudt.org/schema/qudt	Quantities, Units, Dimensions and Types
rdaa	http://rdaregistry.info/Elements/a/	RDA Agent properties
rdac	http://rdaregistry.info/Elements/c/	RDA Classes
rdae	http://rdaregistry.info/Elements/e/	RDA Expression Properties
rdai	http://rdaregistry.info/Elements/i/	RDA Item Properties
rdam	http://rdaregistry.info/Elements/m/	RDA Manifestation Properties
rdau	http://rdaregistry.info/Elements/u/	RDA Unconstrained Properties
rdaw	http://rdaregistry.info/Elements/w/	RDA Work Properties
rdfa	http://www.w3.org/ns/rdfa#	RDFA specification
rdfg	http://www.w3.org/2004/03/trix/rdfg-1/	RDF Graph
rel	http://purl.org/vocab/relationship/	A vocabulary for describing relationships between people

rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
schema	http://schema.org/	Schema.org (converted to OWL by TopQuadrant)
scovo	http://purl.org/NET/scovo#	The Statistical Core Vocabulary (SCOVO)
sim	http://purl.org/ontology/similarity/	The Similarity Ontology
sioc	http://rdfs.org/sioc/ns#	Semantically Interlinked Online Communities
sioctypes	http://rdfs.org/sioc/types#	SIOC Types Ontology
skos	http://www.w3.org/2004/02/skos/core#	SKOS Vocabulary
ssn	http://www.w3.org/2005/Incuba-tor/ssn/ssnx/ssn	Semantic Sensor Network Ontology
time	http://www.w3.org/2006/time#	An OWL Ontology of Time (OWL-Time)
timezone	http://www.w3.org/2006/timezone#	A time zone ontology
ubench	http://swat.cse.lehigh.edu/onto/univ-bench.owl#	An university ontology for benchmark tests
vaem	http://www.linkedmodel.org/schema/vaem#	Vocabulary for Attaching Essential Metadata
vann	http://purl.org/vocab/vann/	Vocabulary for annotating vocabulary descriptions
vcard	http://www.w3.org/2006/vcard/ns#	Vcard vocabulary
voaf	http://purl.org/vocommons/voaf#	Vocabulary of a Friend
void	http://rdfs.org/ns/void#	Vocabulary of Interlinked Datasets (VoID)
vs	http://www.w3.org/2003/06/sw-vocab-sta-tus/ns#	SemWeb Vocab Status ontology
wdrs	http://www.w3.org/2007/05/powder-s#	POWDER-S Vocabulary
xhv	http://www.w3.org/1999/xhtml/vocab#	XHTML specification

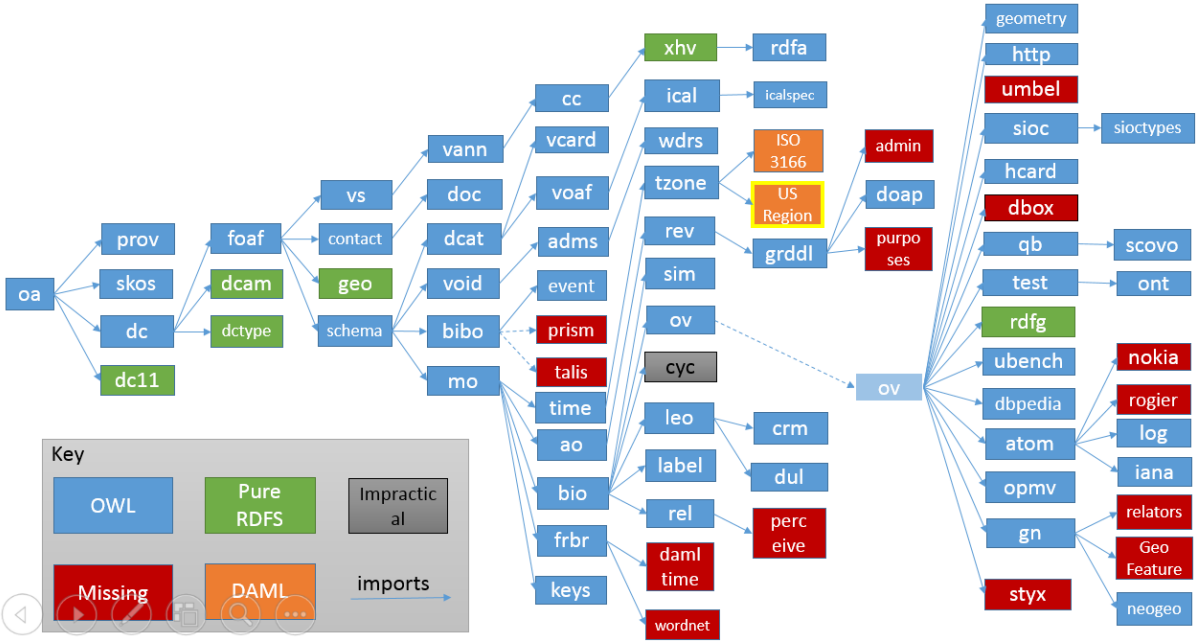

Figure 2 Open Annotation dependency tree of linked data vocabularies and ontologies

Ontology	Missing (or unavailable) Dependencies
atom	property: http://eulersharp.sourceforge.net/2004/04test/rogier#productProperty (1 use) atom:scheme rdfs:range http://sw.nokia.com/WebArch-1/InformationResource rdfs:subPropertyOf http://sw.nokia.com/WebArch-1/representation atom:src rdfs:range http://sw.nokia.com/WebArch-1/InformationResource _:atom31 rdf:type file:///Users/hjs/Programming/sommer/www/atom/2006-06-06/AtomOwl.n3#update _:atom37 rdf:type file:///Users/hjs/Programming/sommer/www/atom/2006-06-06/AtomOwl.n3#rel
doap	doap:Project rdfs:subClassOf http://xmlns.com/wordnet/1.6/Project
frbr	frbr:Work rdfs:subClassOf http://xmlns.com/wordnet/1.6/Work~2 frbr:Event rdfs:subClassOf http://www.isi.edu/~pan/damltime/time-entry.owl#Event
gn	gn:Feature owl:equivalentClass http://www.mindswap.org/2003/owl/geo/geoFeatures20040307.owl#GeographicFeature
grddl	Properties: http://www.rddl.org/purposes#normative-reference (3 uses) & http://webns.net/mvcb/generatorAgent (1 use)
neogeo	http://geovocab.org/spatial owl:imports http://geovocab.org/mappings/spatial
qudt	VOAG ontology only retrievable as invalid turtle file qudt: http://voag.linkedmodel.org/schema/voag#withAttributionTo qudt:NASA-ARC-Attribution qudt: http://voag.linkedmodel.org/schema/voag#hasLicenseType voag:CC-SHAREALIKE_3PT0-US
rda*	All RDA ontologies use terms from missing ontologies http://metadataregistry.org/uri/profile/regap/ and http://metadataregistry.org/uri/profile/rdakit/
timezone	timezone owl:imports http://www.daml.org/2001/09/countries/iso-3166-ont timezone owl:imports http://www.daml.ri.cmu.edu/ont/USRegionState.daml
dbox	Ontology is empty – contains no classes
cwork	http://www.bbc.co.uk/ontologies/tagging/ ontology does not exist cwork:tag rdfs:range http://www.bbc.co.uk/ontologies/tagging/TagConcept cwork:about rdfs:range http://www.bbc.co.uk/ontologies/tagging/TagConcept cwork:mentions rdfs:range http://www.bbc.co.uk/ontologies/tagging/TagConcept

Table 7: References to missing or unavailable dependencies detected

6. Validation Results

Our analysis of the 91 ontologies revealed that 30 ontologies (33%) contained ontology hijacking violations (making assertions about entities defined in other ontologies with global scope). 11 ontologies contained dependencies on a total of 14 missing ontologies (12%). 3 ontologies contained basic errors that were categorized as typos (3.3%). 15 ontologies (16.5%) contained statements that are illegal in OWL DL due to them being impredicative – predicating over classes or properties which is illegal in first order logic entirely - and basic misuses of language constructs (e.g subclassing owl:differentFrom and expecting its semantics to be retained). One ontology (1%) contained both property and class cycles – and in both cases manual analysis revealed that they were, as anticipated, highly likely to be the result of specification errors rather than obtuse ways of defining a single class or property. The detailed validation results are presented in the tables below.

6.1. References to Missing Ontologies

As is to be expected in the evolving Web of Data a number of the referenced ontologies were no longer available (at least they are not currently available at the advertised URL and we were unable to find them elsewhere) - Table 7. The linked data community should be aware of the implication of this for linked data quality – if schema specifications are going to be rendered incomplete due to changes in the availability of imported ontologies or terms then it places a limit on the degree of validation that can be performed – terms from such vocabularies become simple untyped variable names with zero semantics associated with them.

6.2. Ontology hijacking

A widespread pattern observed in the ontologies under analysis was the presence of assertions designed to support interoperability of ontologies. For example, a very common pattern was to specify that certain properties from imported ontologies were defined to be of type owl:AnnotationProperty – to allow them to be processed by standard OWL tools which do not know how to deal with properties defined as rdf:Property.

The basic problem with this pattern is that this amounts to non-coordinated interoperability on a library scope – each ontology attempts to handle interoperability for its own scope, but when these ontologies are combined together, each piecemeal attempt at interoperability is combined into a common model and the union of these piecemeal attempts at library level interoperability without any facilities for modularity leads to inconsistency.

The second major category of ontology hijacking observed in the data are illegal assertions that serve to silently kill error reporting in tools. For example the assertion: rdfs:Class a owl:Class is used in two separate ontologies – it declares that an RDFS class is an instance of an OWL class – an interpretation that is not true under OWL DL or Full but it manages to successfully silence error checking in a number of tools. These type of assertions are particularly unwise because they make the knowledge model inconsistent. They also break the robustness principle by deliberately producing malformed specifications rather than compensating for real-world variation and noise at input.

Finally, in a certain number of cases, ontologies knowingly and explicitly change other ontologies for convenience in utilizing external class definitions. This type of usage is most pointedly described in the bibo ontology:

```
dc:Agent a owl:Class ; owl:equivalentClass
foaf:Agent ;
```

An editorial note in the ontology states: *"BIBO assert that a dcterms:Agent is an equivalent class to foaf:Agent. This means that all the individuals belonging to the foaf:Agent class also belongs to the dcterms:Agent class. This way, dcterms:contributor can be used on foaf:Person, foaf:Organization, foaf:Agent and foaf:Group. Even if this link is not done in neither the FOAF nor the DCTERMS ontologies this is a wide spread fact that is asserted by BIBO."* In such cases it would be more appropriate to use local sub-classing to achieve the equivalent effect without over-writing the definitions in external namespaces.

Table 8: Ontology hijacking violations detected

Ontology	Count	Third party ontologies altered (number of entities altered)
atom	5	iana
bibo	50	rdf (3), rdfs (1), owl (2), dc (19), skos (6), vs (1), event (7), foaf (11)
crm	10	vann (2), dc (3), cc (1), label (1), skos (3)
event	8	dc11 (3), foaf (3), geo (1), vs (1)
foaf	10	owl (1), rdfs (1), dc11 (3), vs (1), geo (1), skos (1), wot (2 – only use)
frbr	32	rdf (1), foaf (3), dc (5), dc11 (7), vann (3), skos (1), cc (11), geo (1)
geometry	3	rdfs (2), dc11 (1)
gn	3	foaf (1), skos (2)
gr	19	owl (1), schema (10), dc11 (5), dc (1), foaf (2)
grddl	1	owl
http	2	rdfs (1), xsd (1)
icalspec	2	xsd
infreal	10	owl (1), rdfs (3), cpa (6)
irw	3	owl (1), infreal (2)
leo	4	crm (3), event (1)
lode	15	leo (11), crm (3), event (1)
mo	1	vs
opmv	6	owl (1), time (5)
prov	6	owl (2), rdfs (4)
prv	33	dc (7), prov(10), infreal (1), foaf (7), wot (4), xhv (2), irw (2)
qudt	8	skos (2), dc11 (6)
rel	1	foaf
rev	9	rdfs (2), dc11 (3), foaf (2), vs (2)
sim	5	owl (1), dc (2), vs (1), foaf (1)
sioc	10	dc (5), foaf (5)
sioctypes	2	skos (1), sioc (1)
ssn	39	rdfs (4), dc11 (6), dc (2), cc (1), dul (26)

time	1	timezone
vaem	12	owl (1), dc (11)

6.3. Typos

Three ontology were found to contain basic errors which were interpreted as typos – the predicate rdfs:ramge appears twice in contact (rather than rdfs:range). In dcat, the property name foaf:Page is used, whereas foaf:page (without capitalization) is the correct property name, while in nrl, 3 incorrect URLs are used to refer to classes and properties in rdfg (the correct URLs use '/' rather than '#' as an element prefix). The presence of such errors in long established and public ontologies highlights the lack of tool support for ontology validation – they are simple and obvious errors but they will not be identified by standard OWL reasoners.

6.4. Impredicativity / misuse of language constructs

Since OWL DL is a first order theory, it is not possible to quantify over classes and predicates. Yet no such restriction exists in RDF. This leads to a number of problems when using OWL ontologies which reference RDF ontologies which make use of higher-order and impredicative features.

In the very widely used dc ontology, the rdf:type relation is given a range of rdfs:Class. This is immediately problematic as rdfs:Class is the class of all classes and such impredicative statements cannot be made in OWL DL but are dangerous regardless, due to the very real threat of paradox. Similarly the rdf:subClassOf relation is used to derive a subclass of the class of classes. This again is higher order reasoning, without any guarantee of predicativity.

In skos we see the use rdf:List as a range, but rdf:List is an internal syntactic element of OWL. Free mixing of rdf:first and rdf:next would leave reasoners unable to distinguish what is intended as a property and what is intended to be syntax of the language itself. While this problem has been described thoroughly [11], it also has not been stamped out in the wild, and skos is a very widely used ontology purporting to be OWL.

Table 9: Typos detected

Ontology	**Typos (underlined)**
contact	contact:assistant rdfs:ramge foaf:Agent contact:participant rdfs:ramge foaf:Agent
dcat	dcat:landingPage rdfs:subPropertyOf foaf:Page
nrl	nrl:subGraphOf rdfs:subPropertyOf http://www.w3.org/2004/03/trix/rdfg-1#subGraphOf nrl:Graph rdfs:subClassOf http://www.w3.org/2004/03/trix/rdfg-1#Graph nrl:equivalentGraph rdfs:subPropertyOf http://www.w3.org/2004/03/trix/rdfg-1#equivalentGraph

In gn, log, void, qb, wdrs, atom, voaf we see the very common use of higher order logic, with subclassing of class, properties, and assignation of ranges over properties and classes. In most of these cases the statements were probably unnecessary. However higher order reasoning may sometimes be useful and we will discuss later how such things can be achieved without stepping into undecidability.

In atom there is an even more unusual metalogical statement, making a statement about statements themselves! Without some sort of stratification such logic is dubious at best. Atom additionally makes use of inference facilities that are not themselves part of OWL. Utilizing ontologies of this form requires a tool chain which is capable of making these inferences, which is something which is not widely available.

Table 10: Instances of impredicativity/misuse of reserved language constructs detected

Vocab	**Triple(s)**	**Error Description**
dc	dc:type rdfs:range rdfs:Class ;	Predicating over class
dc	dc:AgentClass rdfs:subClassOf rdfs:Class	Overriding basic language construct
skos	skos:memberList rdfs:range rdf:List ;	rdf:List is an internal structural element of OWL – it can't be used directly
grddl	grddl:TransformationProperty rdfs:subClassOf owl:FunctionalProperty ;	Higher order use of the rdfs:subClassOf relation
wdrs	wdrs:Document rdfs:subClassOf owl:Ontology .	Higher order use of the rdfs:subClassOf relation

rel	rel:friendOf rdfs:subPropertyOf owl:differentFrom (32 times)	Higher order use of the rdfs:subClassOf relation
atom	atom:RelationType rdfs:subClassOf owl:ObjectProperty .	Higher order use of the rdfs:subClassOf relation
atom	atom:Link rdfs:subClassOf rdf:Statement	Creating subclasses of a higher order feature
atom	atom:rel rdfs:subPropertyOf rdf:predicate	Creating subclasses of a higher order feature
atom	atom:subject rdfs:subPropertyOf rdf:subject	Creating subclasses of a higher order feature
atom	atom:to rdfs:subPropertyOf rdf:object	Creating subclasses of a higher order feature
bio	bio:differentFrom rdfs:subPropertyOf owl:differentFrom (15 times)	Higher order use of the rdfs:subClassOf relation
gn	gn:featureClass rdfs:subPropertyOf dc:type ;	Using impredicative property from dc
log	log:definitiveDocument rdfs:domain rdf:Property	Predicating over class of properties
log	log:definitiveService rdfs:domain rdf:Property ;	Predicating over class of properties
void	void:linkPredicate rdfs:range rdf:Property	Predicating over class of properties
void	void:property rdfs:range rdf:Property	Predicating over class of properties
voaf	voaf:occurrences a owl:objectProperty, rdfs:range xsd:integer	Mismatch between objectProperty and literal range type
qb	qb:parentChildProperty rdfs:range rdf:Property	Predicating over class of properties
qb	qb:ComponentProperty rdfs:subClassOf rdf:Property	Higher order use of the rdfs:subClassOf relation
bbcpro	bbcpro:transitions rdfs:range rdf:Property	Predicating over class of properties
nrl	nrl:cardinality rdfs:domain rdf:Property nrl:maxCardinality rdfs:domain rdf:Property nrl:minCardinality rdfs:domain rdf:Property nrl:inverseProperty rdfs:domain rdf:Property nrl:inverseProperty rdfs:range rdf:Property	Predicating over class of properties
nrl	nrl:NonDefiningProperty rdfs:subClassOf rdfs:Property	Higher order use of the rdfs:subClassOf relation
qudt	qudt:QuantityKindCategory rdfs:subClassOf owl:Class	Higher order use of the rdfs:subClassOf relation

6.5. Property / Class Cycles

Table 11 presents the class or property cycles detected in the crm ontology. The first example, asserts that a legal body is equivalent to a group, which seems highly questionable, though it would require the crm authors to confirm. The second looks more likely, but still questionable, where they establishing an equivalence between "bearing a feature" and "being composed of".

We have also noticed that many statements of equivalence were between classes in different ontologies, establishing a link between an element in one place, and that in another. However, these equivalences were often coupled with additional qualifications. Such behavior completely negates the capacity to use linked data in an interoperable fashion, as the original publisher of the ontologies data may very well have instance data which is deemed invalid when read, by the second publisher, and vice versa. This "ontology hijacking" [5] should be highly discouraged.

6.6. Comparison to OOPS!

The 50 most commonly used vocabularies were also analysed with OOPS! (OntOlogy Pitfall Scanner) [27] for comparison (although four failed to load).A this tool analyzes ontologies in isolation without loading any dependencies, the dependent ontologies analyzed by Dacura were not included. There is very little intersection between the classes of violations / pitfalls identified between the two systems because OOPS! is primarily a syntax scanner and it does not attempt to incorporate dependent ontologies and combine them into a unified model, nor does it apply any significant reasoning. However, OOPS! does check for several additional types of best-practice violations that are not considered to be violations from Dacura's point of view. For example, the P08 missing annotations code produced by OOPS! reports cases where classes or properties are missing labels – while this is a useful check, there is nothing illegal about such missing elements and they thus do not cause Dacura to reject the ontology. The only areas where Dacura and OOPS! overlap is in the identification of absent domain /

range assertions for properties and secondly in the identification of untyped classes and properties. Dacura also checks for such violations, however, in Dacura they are considered to be strictly informational messages as in many cases, such missing assertions are consistent with best practice, e.g. when the domain or range is specified in a super-property. In such cases, OOPS! violations will in fact be incorrect because it does not attempt to load super-properties and respecifying the domain or range in a sub-property duplicates information which complicates schema change management. Table 12 shows the results of testing the covered ontologies and vocabularies with OOPS!

Table 11: Property/Class cycles detected in crm ontology

Triple(s)	**Problem**
crm:E40_Legal_Body rdfs:subClassOf crm:E74_Group crm:E74_Group rdfs:subClassOf ns1:E40_Legal_body	Cycle in class hierarchy
crm:P46_is_composed_of rdfs:subPropertyOf crm:P56_bears_feature; crm:P56_bears_feature rdfs:subPropertyOf crm:P46_is_composed_of	Cycle in property hierarchy

Table 12: Results returned by OOPS! Pitfall Scanner – numbers indicate pitfall count per OOPS! code

	OOPS! Pitfall Codes – for code meanings see http://oops.linkeddata.es/catalogue.jsp																								
Ontology	2	4	7	8	10	11	12	13	19	20	21	22	23	24	25	26	30	31	32	34	35	36	38	39	40	41
adms						9		11												6	3					1
basic						4		2									2			1						
bbccor																						1				
bbcpro																						1				
bibo		1		70	1	40		48				1		1			2				1					1
bio																									1	
cc																							1	1		1
cpa																						1				1
dbpedia																							1	1		
dcat				1	1			10		2										6	4					1
dc11																										
dc																				20	16		1	1		1
dctype																							1	1		1
doap		2		2		7	2	27				1								9	1					
dul				16		3				1				20	9	9	4					1				1
event		1		63		17		14				1		2			1					1				1
foaf		2		2		7	2	27				1								4	1					1
frbr		9		69		11		6														1			8	
geo																				1	1		1	1		1
gn				21	1	13	1	15												7	3				2	1
gr			3	11		6		43				1		2	2	2	2	2								
gsp				69		13		53											13	2						1
leo				4	1			7				1								10	12				2	

mo																								1	
nrl																			9	2		1	1		1
oa		1		1	1	14		14											1	2					1
obo	Failed to load																								
opmv				1				13											2						1
org				3		4		14		6									6	4				1	
prov				23		2		42	2																1
prv		13		35		27																			
qb		1		5		6		17											3	1	1				
qudt		1	10	296		107	5	32				1		2	1	1			1					8	
rdac																			4			1			1
rdaw	Failed to load																								
rdfg																			1			1			1
schema																			10	9		1	1		1
scovo		3			1																			5	1
sim	Failed to load																								
sioc		3		12		41	2	18				1							2						1
skos				16		13		5																	1
ssn	1			23		44		27		1				17	4	4	6	1						1	
time				53		7		10						2									1		1
vann	Failed to load																								
vcard	2	4		30		75		50			1	1					1	1							1
voaf		1		1	1	5		14											3	2				1	
void		1			1														1					5	1
vs																								2	1
wdrs						1													3				1		1

7. Recommendations for Correcting Problems in Linked Data Schemata

Given our experiences in constructing the DQS and the experimental analysis performed of real world linked data schemata we offer the following recommendations for improving best practice in linked data vocabulary design.

7.1. Metareasoning with rdf:List

Ontologies in OWL cannot use the underlying syntactic elements of rdf:List within the logic of the ontology, as is done in SKOS. The appropriate way to deal with this problem is to have a drop in replacement for RDF collections written in OWL such that there is no syntactic/logical mixing. There have been some list ontologies constructed such as the Ordered List Ontology[6], CO-ODE List Ontology[7], however what is needed is a drop in replacement for RDF collections in general. Bags are trivial to construct in OWL, and both

[6] https://smiy.wordpress.com/2010/07/15/the-ordered-list-ontology/

[7] http://owl.cs.manchester.ac.uk/wp-content/uploads/2015/07/list.owl_.txt

ordered lists and indexed sequences have also been demonstrated, so creating such an ontology is more a collation task than an ontology engineering one. Migrating current OWL ontologies to use such a drop in replacement would be a relatively minor task and would allow them to be compliant OWL DL.

7.2. Impredication and Higher order features

The impredicative and higher order features of RDF are used by 15 of the top 50 ontologies (including their dependencies) and hence it can be considered a common problem. Supporting such behavior does not require abandoning soundness or allowing paradox. Type theory, going back to Russell, developed techniques to avoid impredicative paradoxes through the use of some form of stratification, which could be used to extend OWL DL. The complexity or indeed decidability of such an extension remains to be explored.

A lot of the uses of predication over types (eg in dc) are useful and have known solutions, e.g. [42], [43] so it is strange to reject it as outside OWL DL. This is the reason naïve set theory is inconsistent. Punning provides some useful ways of providing information about classes and properties. However, this does not enable the same logical power which is available through stratified predication where reasoning can be extended to the metalogical level.

7.3. Equivalence and Hijacking

From the ontologies surveyed, it appears that equivalence within a given ontology is rarely needed. If a class is the same as another class, it seems unlikely to be the case that the ontology designer does not know it. If two classes are indeed the same, it is best to combine the definitions of the classes into a single class, which improves referential transparency and simplifies ontology management. If two names are needed, simply assigning more than one rdfs:label is recommended as a better solution.

However, there is the further use of identification of one class with that of another ontology. Such identification of classes with other ontologies leads to the question of why one would simply not use the class name from the alternative ontology unless one wants to actually hijack the class for extension? And if it is the later, then it seems unfair that the contract be entirely one sided, as any published linked data which comes from the ontology will no longer have the same meaning as that given in the original ontology.

One potential answer to this problem is that ontologies which intend to coordinate, and actually mean to be equivalent, utilise subclassing in either direction. So for instance, instead of saying:

```
ex:Tome owl:EquivalentClass library:Book
```

One could say, in the ex and library ontologies respectively:

```
ex:Tome rdfs:subClassOf library:Book
library:Book rdfs:subClassOf ex:Tome
```

In this scenario, collaboration between ontology designers would be required, such that hijacking was less of a concern.

Where it is necessary to make ontologies backward compatible with existing tools, a custom ontology should be constructed and all interoperability assertions should be placed within it and then imported. Beyond such cases, Ontology hijacking should be avoided in all cases – just like when using external libraries in software engineering, importing ontologies should not have side effects on other ontologies. We propose a general design principle that importing ontologies should have no side effects.

8. Conclusions and Future Work

We have shown that is effective to pursue a reasoner-based approach to detect logical or syntactic errors in linked data schemata based on unified logical models. We have made a first study of the prevalence of errors in schema errors in the Web of Data by analyzing 91 common vocabulary or ontology specifications. Our validation detected a total of 6 typos, 14 missing or unavailable ontologies, 73 language level errors, 310 instances of ontology namespace violations and 2 class cycles which we believe to be errors. Although our analysis is not complete – there are undoubtedly further errors which we have not detected – all of these errors represent genuine problems with the analyzed ontologies and there are no other tools available which can identify more than a small fraction of them.

Our analysis began with the practical concern of using Open Annotation (OA) as infrastructure for our own ontology development. After producing a software tool-chain which included ontology management and reasoning, we were able to proceed to testing of our ontology over OA and all of the ontologies which it made reference to and from there to extend our survey to all of the most commonly used 50 ontologies and all of their dependencies. The results of our survey give valuable information about the state of ontology

development, the relative lack of interoperability including the free mixing of ontological frameworks which are logically incompatible, and the fact that tool-chain development is at a very low level since many problems which we found would otherwise have been spotted already.

We make a number of recommendations regarding how to deal with the realities of ontologies as they currently exist, and how to use them in conjunction with reasoning tool-chains.

We also note the fairly widespread use of higher order features used for meta-modelling, and suggest a way to include such features in a sound fashion free of paradoxes. We hope to explore the consequences of adding stratification to OWL DL and the decidability and complexity consequences thereof in the future.

The utilization of rdf:List in OWL ontologies really has to be eliminated as it leads to incoherence and the incapacity to reason. In the future, we hope to develop a drop in replacement ontology for rdf collections defined in OWL DL exclusively.

We will be extending our reasoner to include a larger fragment of OWL DL. Our system has already proved useful in finding errors and contains the majority of OWL descriptions which we found in the ontologies explored. A larger fragment should improve the usefulness as it extends the reasoning facility to a greater class of ontologies. Further, we will be testing our reasoner against ontologies which have extant instance data, and this is likely to reveal more problems than the ones detailed here which are exclusively at the schema level.

Acknowledgement

The authors would like to thank Odhran Gavin at TCD for his assistance with manuscript preparation and editing.

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644055, the ALIGNED project (www.aligned-project.eu) and from the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded by the European Regional Development Fund.

References

[1] C. Bizer, T. Heath and T. Berners-Lee (2009) Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, Vol. 5(3), Pages 1-22. DOI: 10.4018/jswis.2009081901

[2] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, S. Decker, An empirical survey of Linked Data conformance, *Journal of Web Semantics*, 14, 14–44, 2012

[3] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):7-26, 2003

[4] G. T. Heineman and W. T. Councill (Eds.). 2001. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[5] Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, Axel Polleres, Weaving the pedantic web, in: *3rd International Workshop on Linked Data on the Web*, LDOW2010, April 2010.

[6] Kevin C. Feeney, Declan O'Sullivan, Wei Tai, and Rob Brennan. Improving curated Web-Data quality with structured harvesting and assessment. *International Journal on Semantic Web and Information Systems* 10(2):35-62, April 2014

[7] Hepp, M., Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies, in *Internet Computing, IEEE*, vol.11, no.1, pp.90-96, Jan.-Feb. 2007.

[8] ISO 9001:2015 Quality Management Systems - Requirements, ISO/TC 176, Quality management and quality assurance, Subcommittee SC 2, Quality systems. 2015

[9] John Daintith, *A Dictionary of Computing* 2004, Oxford University Press 2004.

[10] Axel Polleres, Aidan Hogan, Renaud Delbru, and Jurgen Umbrich, (2013) RDFS and OWL Reasoning for Linked Data, Reasoning Web Summer School, Mannheim, 91–149, Springer, 2013. Available at: https://www.deri.ie/sites/default/files/publications/paper.pdf

[11] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The Making of a Web Ontology Language. *J. of Web Semantics*, 1(1):7-26, 2003

[12] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. 2008. OWL 2: The next step for OWL *Journal of Web Semantics.* 6, 4 (November 2008), 309-322.

[13] P. F. Patel-Schneider, D. Fensel, Layering the Semantic Web:Problems and Directions, *First International Semantic Web Conference* (ISWC2002), Sardinia, Italy, June 2002.

[14] Z. Luo, An Extended Calculus of Constructions, PhD. Thesis, University of Edinburgh, 1990 Available at: http://www.lfcs.inf.ed.ac.uk/reports/90/ECS-LFCS-90-118/ECS-LFCS-90-118.pdf

[15] Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, Alan Ruttenberg (2009), MIREOT: the Minimum Information to Reference an External Ontology Term, Proc. First International Conference on Biomedical Ontology, 26 July 2009

[16] Gruber, Thomas Robert (1992). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies* **43**: 907–928.

[17] Matthem Horridge, Sean Bechhofer, The OWL API: A Java API for OWL Ontologies, *Semantic Web Journal* 2(1), Special Issue on Semantic Web Tools and Systems, pp. 11-21, 2011

[18] B. Motik, P. F. Patel-Schneider, B. C. Grau (Eds.) OWL 2 Web Ontology Language - Direct Semantics (Second Edition), W3C Recommendation 11 December 2012, Available at:

http://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/

[19] Gavin Mendel-Gleason, Kevin Feeney, Rob Brennan (2015) Ontology Consistency and Instance Checking for Real World Linked Data, *Proceedings of the 2nd Workshop on Linked Data Quality* co-located with 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, June 1, 2015. Available at: http://ceur-ws.org/Vol-1376/LDQ2015_paper_03.pdf

[20] Jan Wielemaker, Wouter Beek, Michiel Hildebrand, Jacco van Ossenbruggen, (2015) ClioPatria: A SWI-Prolog infrastructure for the Semantic Web, Semantic Web, vol. Preprint, no. Preprint, pp. 1-13, 2015, DOI: 10.3233/SW-150191

[21] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd international conference on World wide web* (WWW '14). ACM, New York, NY, USA, 747-758. DOI=http://dx.doi.org/10.1145/2566486.2568002

[22] E. Sirin and J. Tao. Towards integrity constraints in owl. In Proceedings of the Workshop on OWL:Experiences and Directions, OWLED, 2009.

[23] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann and Sören Auer. (2015). Quality assessment for linked data: a survey. Semantic Web. vol. Preprint, no. Preprint, pp. 1-31, 2015 DOI: 10.3233/SW-150175

[24] C. Furber and M. Hepp. Using sparql and spin for data quality management on the semantic web. In W. Abramowicz and R. Tolksdorf, editors, BIS, volume 47 of Lecture Notes in Business Information Processing , pages 35-46. Springer, 2010.

[25] Peter F. Patel-Schneider, (2015)Using Description Logics for RDF Constraint Checking and Closed-World Recognition, Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI-2015, January 25–30, 2015, Austin Texas, USA. Available at: http://arxiv.org/abs/1411.4156v2

[26] Jeremy Debattista, Christoph Lange, Sören Auer: Luzzu - A Framework for Linked Data Quality Assessment. *International Semantic Web Conference* (Posters & Demos) 2015

[27] María Poveda-Villalón, Asunción Gómez-Pérez and Mari Carmen Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation IJSWIS 10.2 (2014): 7-34. Web. 15 Mar. 2016. doi:10.4018/ijswis.2014040102

[28] P. F. Patel-Schneider, D. Fensel, Layering the Semantic Web:Problems and Directions, First International Semantic Web Conference (ISWC2002), Sardinia, Italy, June 2002

[29] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The Making of a Web Ontology Language. *J. of Web Semantics*, 1(1):7-26, 2003

[30] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. 2008. OWL 2: The next step for OWL. J. Web Semantics. 6, 4 (November 2008), 309-322. DOI=http://dx.doi.org/10.1016/j.websem.2008.05.001

[31] Mendes, P., Muhleisen, H., and Bizer, C. Sieve: Linked data quality assessment and fusion. In LWDM (March 2012).

[32] Chris Bizer. Quality-Driven Information Filtering in the Context of Web-Based Information Systems. PhD thesis, Freie Universität Berlin, March 2007.

[33] Tim Berners-Lee, James Hendler and Ora Lassila (2001). The Semantic Web. *Scientific American*, May 2001, p. 29-37.

[34] Atanas Kiryakov, Damyan Ognyanoff, Ruslan Velkov, Zdravko Tashev, Ivan Peikov (2009) LDSR: Materialized Reasonable View to the Web of Linked Data, *OWL: Experiences and Directions workshop* (OWLED). Chantilly, Virginia, USA, 2009.

[35] Gianluca Demartini, Djellel Eddine Difallah, Philippe Cudré-Mauroux, Large-scale linked data integration using probabilistic reasoning and crowdsourcing, The VLDB Journal (2013) 22:665–687, DOI 10.1007/s00778-013-0324-z

[36] Jan Wielemaker, Wouter Beek, Michiel Hildebrand, Jacco van Ossenbruggen, (2015) ClioPatria: A SWI-Prolog infrastructure for the Semantic Web, Semantic Web, vol. Preprint, no. Preprint, pp. 1-13, 2015, DOI: 10.3233/SW-150191

[37] Turchin, Peter; Brennan, Rob; Currie, Thomas; Feeney, Kevin; Francois, Pieter; Hoyer, Daniel; et al.(2015). Seshat: The Global History Databank. *Cliodynamics: The Journal of Quantitative History and Cultural Evolution*, 6(1). irows_cliodynamics_27917. Retrieved from: http://escholarship.org/uc/item/9qx38718

[38] Pierre-Yves Vandenbussche, Ghislain A. Atemezing, Maria Poveda, Bernard Vatant, Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web, in press, http://www.semantic-web-journal.net/content/linked-open-vocabularies-lov-gateway-reusable-semantic-vocabularies-web-1

[39] Max Schmachtenberg, Christian Bizer, and Heiko Paulheim, Adoption of the Linked Data Best Practices in Different Topical Domains, in P. Mika et al. (Eds.) ISWC 2014, Part I, LNCS 8796, pp. 245–260, 2014.

[40] Aidan Hogan, Jürgen Umbrich, Andreas Harth, Richard Cyganiak, Axel Polleres, Stefan Decker, An empirical survey of Linked Data conformance, Web Semantics: Science, Services and Agents on the World Wide Web 14 (2012) 14–44

[41] Robert Sanderson, Paolo Ciccarese, Herbert Van de Sompel (Eds.), (2013) Open Annotation Data Model, W3C Community Draft, 8 February 2013, Available at: http://www.openannotation.org/spec/core/20130208/

[42] Z. Luo, An Extended Calculus of Constructions, PhD. Thesis, University of Edinburgh, 1990 Available at: http://www.lfcs.inf.ed.ac.uk/reports/90/ECS-LFCS-90-118/ECS-LFCS-90-118.pdf

[43] Bourbaki, Nicolas (1972). "Univers". In Michael Artin, Alexandre Grothendieck, Jean-Louis Verdier, eds. Séminaire de Géométrie Algébrique du Bois Marie – 1963-64 – Théorie des topos et cohomologie étale des schémas – (SGA 4) – vol. 1 (Lecture notes in mathematics 269) (in French). Berlin; New York: Springer-Verlag. pp. 185–217.