The OWL Full/DL gap in the field

Nicolas Matentzoglu and Bijan Parsia

The University of Manchester
Oxford Road, Manchester, M13 9PL, UK
{bparsia,matentzn}@cs.manchester.ac.uk

Abstract. OWL 2 Full remains a "catch all" language in the sense that it treats as well formed any legal RDF graph. In contrast, OWL 2 DL, and its sub profiles, exclude large classes of RDF graphs as malformed and thus meaningless. Many ontology documents on the web appear to fall under OWL Full. However, not all ways of being OWL Full are indicative of modelling intent. In this paper, we look at the prevalence of OWL 2 DL violations in a large corpus of ontologies gathered from the Web[1]. We present a new classification of violations and analysis of those that are safely repairable. We find that a high preponderance (69%) of violations are repairable and are, indeed, mainly "mere" *declaration* failures. This suggests for example that declarations as a conformance criterion should be rethought.

Keywords: OWL, Ontologies, OWL profiles, repair

1 Introduction

Both the OWL 1 and OWL 2 standards define *families* of languages[2] with varying expressivity. One goal of the OWL 2 working group was to improve on the profile distinctions of OWL 1. In OWL 1, there were three profiles:

1. OWL Lite, which was intended to be an easy to learn, easy to implement, computationally tractable version of OWL. Unfortunately, it's expressive power was to encode the Description Logic $\mathcal{SHIF}$ which precluded its being easy to implement and computationally tractability ($\mathcal{SHIF}$'s key decision problems are in EXPTIME). Furthermore, due to the (ineffectual) contortions of the grammar, it was quite difficult to model, especially if one was trying to avoid "hard" constructions.
2. OWL DL, which was close to a known maximal DL ($\mathcal{SHOIN}$). $\mathcal{SHOIN}$ was shown to be decidable, and "reasonable" implementation techniques were known. Unfortunately, an effective, goal-directed algorithm for deciding $\mathcal{SHOIN}$ satisfiability was not discovered until Horrocks et al. 2007 [3], although implementations followed swiftly afterwards.

[1] Note: It is of course the case that the "violations" of the OWL 2 DL syntactic constraints might be deliberate and desired. That an ontology violates those conditions is not, *ipso facto*, a problem with the ontology. It still might be desirable to generate an approximate OWL DL ontology from an OWL Full ontology.

[2] Known as *species* in OWL 1 and *profiles* in OWL 2. We will use *profile* exclusively.

3. OWL Full, which was in terms of the RDF syntax maximally and a "straightforward" extension of the idiosyncratic semantics of RDF. (In contrast, $\mathcal{SHOIN}$ is a notational variant of a fragment of first order logic extended with counting quantifiers.)

Thus, none of the OWL 1 profiles straightforwardly met their desiderata. In particular, the "lite" profile really was not lightweight and the gap between OWL DL and OWL Full was larger than it needed to be in some odd ways. Furthermore, the utility of many features of OWL Full was very unclear and there was evidence that further extensions would be paradoxical [7].

One reason that OWL Light was a failure is that the state of the art in lightweight DLs was relatively impoverished at the time of standardisation. Subsequently, there was a renaissance of research into sub-Boolean Description Logics in the form of the $\mathcal{EL}$ family, the *DL Lite* family, and Description Logic Programs (DLP). Each of these families were well understood, had reasonable implementations, and addressed concrete application spaces. For these languages it was also, demonstrably, much easier to get a non-toy implementation up and running.

OWL 2 introduced a new set of profiles:

1. OWL 2 EL, OWL 2 QL, and OWL 2 RL, which are based on usefully expressive Description Logics with key decisions problems in PTIME. They also were designed in light of known implementation techniques. For example, OWL RL was tuned for forward chaining rule engines and even included a (non-optimal) rule set as a reference implementation. These, collectively, replaced OWL Lite.
2. OWL 2 DL, which was close to a known maximal DL (in this case, $\mathcal{SROIQ}$) for which "reasonable" implementation techniques were known (and already implemented). Furthermore, there were extensions (such as a restricted form of metamodelling known as "punning") to narrow the gap between OWL 2 DL and OWL 2 Full.
3. OWL 2 Full, remains a syntactically maximal, same-semantics extension of RDF. The semantics were altered in a variety of ways to address concerns and generally make them weaker (and thus less prone to contradiction and paradox as well as somewhat easier to work with).

OWL 2 Full remains a "catch all" language in the sense that it treats as well formed any legal RDF graph. In contrast, OWL 2 DL, and sub profiles, exclude large classes of RDF graphs as malformed and thus meaningless. If it was the intention of an ontology author to create an ontology outside of OWL 2 DL, then the existence of such graphs is a problem. However, not all ways of being OWL Full are indicative of modelling intent. For example, missing declarations nominally place an ontology outside OWL 2 DL, but since this almost never has any real effect on the meaning of the ontology, it is undesirable for OWL 2 DL tools to reject such ontologies, or to switch into OWL 2 Full mode.

There is a tradition in the OWL community of "repairing" certain classes of OWL 2 Fullisms (or, to a lesser extent, other profile violations) either automatically or interactively. Such repair strategies can suggests revisions to the profiles to bring OWL 2 DL and OWL 2 Full closer together. In the ideal, they would converge, or, at least, have clearly motivated uses. Applying syntactic patches to OWL Full ontologies has been proposed for example by Bechhofer et al.[1, 9].

Some repairs are lossless or effectively lossless, i.e., there is no entailment (perhaps "of note") lost by converting the OWL 2 Full ontology into an OWL 2 DL ontology. However, some repairs are approximations. For example, repairing an unambiguously inferable declaration is lossless. Dropping a transitivity axiom for a property that appears in a cardinality restriction removes several, almost certainly intended, entailments. Reasoners such as Pellet[8] try to repair unsupported axioms internally, but often, OWL DL violations are dealt with by simply dropping the violating axiom. In the case of an illegal axiom interaction, this could potentially be non-deterministic (dropping *one* of the interacting axioms).

The higher the degree we can expand the class of repairs (or build them into the language) the better. At the moment, users are still caught between access to the bulk of the OWL infrastructure (i.e., staying in OWL 2 DL) or not having to conform to rather complex syntactic conditions which are idiosyncratically enforced (i.e., falling into OWL 2 Full).

In this paper, we look at the prevalence of OWL 2 DL violations in a large corpus of ontologies gathered from the Web. We present a new classification of violations and analysis of which are safely repaired and apply safe repairs to that corpus. We find that a high preponderance (70%) of violations are repairable and, indeed, are mainly "mere" *declaration* failures. This suggests that declarations as a conformance criterion should be rethought.

2 Errors and conformance in OWL

OWL ontology documents can be encoded in a wide variety of syntaxes, but the only one required by the standards is the RDF/XML syntax. Document conformance[3] is thus defined in terms of RDF/XML. We do not deal with errors at the XML or RDF level, which is normal in OWL repair discussions. Thus, if the document is not well-formed XML or does not parse to a legal RDF graph, we do not treat it as any sort of OWL ontology. The only exception here are non-absolute IRIs. While they are strictly illegal (both in the RDF data model and the XML, since not wellformed), most parsers (such as the ones in the OWL API) do process them anyway. In some sense it might be arguable why such violations are attributed to the OWL Full / OWL DL gap, since they are not even OWL Full or RDF(S) strictly speaking. We include them here however, because they are a very common violation, straight forwardly fixable, and would lift a good number of ontologies into the OWL DL profile.

The key conformance distinction we consider is whether a document is OWL 2 Full or OWL 2 DL. According to the conformance spec, every possible RDF/XML

[3] `http://www.w3.org/TR/owl2-conformance/#Document_Conformance`

document encodes an OWL 2 Full ontology, but only those graphs that "survive" the canonical parsing process[4] are OWL 2 DL ontologies. The canonical parsing process is rather complex and includes a scattered set of constraints (mostly gathered in Section 3 of the Structural Specification[5]).

Tools which have defined OWL 2 conformance conditions are entailment checkers and query answering tools (in other words, reasoners). A conforming reasoner that is given documents which do not meet the document conformance conditions (that is, exhibit a parsing failure) must return `Error`. Any attempt to repair the document along the lines discussed here makes the tool non-conforming.

This is reasonable and many reasoners have a flag that allows them to be non-conforming and thus more forgiving of variant input. Furthermore, there is no conformance constraint on ontology editors or preprocessors, so it is easy to imagine pairing such tools with reasoners in order to handle more documents. Of course, given that some repairs are fairly non-deterministic, this will result in hard to cope with variance between such tools. This gives more imperative to either standardising repair and approximation strategies or, at least, making the process better defined. Reasoners with a repair strategy would do well to clearly document it, or, better yet, separate it out so that the repair strategy can be used across reasoners.

3 Classification of Violations

Our direct, application purpose in revisiting OWL 2 DL violations is the development of a large corpus of OWL documents crawled from the Web. This corpus is primarily intended for experimentation with OWL 2 DL tools, esp. reasoner performance. Thus, we want as broad a set of OWL 2 DL documents and are rather more tolerant of meaning-altering changes. We also believe that, given the huge difference in available infrastructure between OWL 2 Full and OWL 2 DL (most importantly sound, complete and terminating reasoning procedures) that keeping one's ontology in OWL 2 Full should be a carefully considered choice which is deliberately made for some tangible reason. It should be noted that while the followings enumeration of violations is intended to be complete (and we do believe it is complete), it should not be taken as guaranteed, since there is no official document listing all violations in a straight forward way, and they do have to be extracted manually from the OWL 2 specification.

We distinguish between four classes of profile violations:

1. Declaration Failure (vio-dec): Entities such as classes and properties have to be declared according to the OWL 2 specification[6], section 5.8.
2. Inherent Violation of the DL Profile (vio-dl): Some axioms are expressible in OWL (legal syntax), but do not have the benefit of clear DL semantics

[4] `http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Canonical_Parsing_of_OWL_2_Ontologies`

[5] `http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Ontologies`

(OWL 2 Full), such as (DL) illegal punnings or (DL) illegal use of non-simple object properties.
3. Misuse of reserved vocabulary (vio-res): IRIs from the reserved vocabulary (apart from some exceptions such as *owl:Thing*) must not be used to identify entities in an axiom.
4. Violation of the OWL Syntax (vio-syn): Axioms or expressions that violate the grammar of OWL 2 fall under this category. For example, some axioms require a specific number of operands and IRIs of the elements in OWL 2 have to be well formed.[6]

In the following, we will discuss these classes of violations in more detail. Numbers in brackets behind violation descriptors indicate section in the OWL 2 specification[6].

3.1 Declaration Failure

Failure of declaration is by far the most common violation. According to the OWL 2 DL specification[6], section 5.8, entities used in axioms have to be declared somewhere. There are five types of missing declarations according to our classification:

1. UseOfUndeclaredClass
2. UseOfUndeclaredAnnotationProperty
3. UseOfUndeclaredObjectProperty
4. UseOfUndeclaredDatatype
5. UseOfUndeclaredDataProperty

3.2 Inherent Violation of the DL Profile

We are distinguishing between 13 different kinds of violations that clearly violate the DL profile and cannot be automatically fixed without in some way or another change the logical structure of the ontology. The restrictions on simple roles are summarised in [6], section 11.8.

1. Illegal use of non-simple properties (11.8)
   (a) UseOfNonSimplePropertyInCardinalityRestriction
   (b) UseOfNonSimplePropertyInIrreflexivePropertyAxiom
   (c) UseOfNonSimplePropertyInObjectHasSelf
   (d) UseOfNonSimplePropertyInAsymmetricObjectPropertyAxiom
   (e) UseOfNonSimplePropertyInInverseFunctionalObjectPropertyAxiom
   (f) UseOfNonSimplePropertyInFunctionalPropertyAxiom

[6] There is a tension in the recommendations. The structural syntax clearly restricts the term *OWL 2 Ontology* to ontologies meeting these basic syntactic criteria. However, this entails, by the reverse translation to RDF, that some RDF graphs are not *OWL 2 Ontologies*. However, the RDF Semantics treats *every* RDF graph as an OWL Full ontology and assigns it a specific meaning.

   (g) UseOfNonSimplePropertyInDisjointPropertiesAxiom
2. UseOfPropertyInChainCausesCycle (11.2)
3. IllegalPunning[7]
4. UseOfTopDataPropertyAsSubPropertyInSubPropertyAxiom (11.2)
5. CycleInDatatypeDefinition (11.2)
6. UseOfBuiltInDatatypeInDatatypeDefinition (11.2)
7. UseOfDefinedDatatypeInDatatypeRestriction (9.4)

3.3 Misuse of reserved vocabulary

A very common example for misusing reserved vocabulary is to create a subproperty of an existing RDF property, such as *rdfs:label*. According to the specification (sections 5.1 - 5.6), most entities that are part of the reserved vocabulary (with a few exceptions, such as rdfs:Literal) cannot be directly used in an axiom.

1. UseOfReservedVocabularyForOntologyIRI (3.1)
2. UseOfReservedVocabularyForObjectPropertyIRI (5.3)
3. UseOfReservedVocabularyForAnnotationPropertyIRI (5.5)
4. UseOfReservedVocabularyForIndividualIRI (5.6)
5. UseOfReservedVocabularyForDataPropertyIRI (5.4)
6. UseOfReservedVocabularyForVersionIRI (3.1)
7. UseOfReservedVocabularyForClassIRI (5.1)
8. UseOfUnknownDatatype (5.2): A datatype that is neither in the OWL 2 datatype map, nor rdfs:Literal, but part of the reserved vocabulary.

3.4 Violation of the OWL Syntax

Some ontologies contain expressions that are not well-formed with respect to the OWL 2 grammar. Parsers could choose to rigorously reject these kinds of ontologies (draconian error handling), for example, the Functional Syntax Parser of the OWL API is very unforgiving with respect to wellformedness. Many parsers, however, such as the OWL/XML parser allow parsing all sorts of broken expressions. The most common violations of this type are the use of non-absolute IRIs for entities in OWL 2, or an insufficient number of operands in certain expressions (unary, binary and n-ary).

1. Non absolute IRIs (2.4)
   (a) OntologyIRINotAbsolute
   (b) OntologyVersionIRINotAbsolute
   (c) UseOfNonAbsoluteIRI
2. Insufficient Operands in Expression
   (a) InsufficientIndividuals (9.6.2,9.6.2)
   (b) InsufficientDataRangeOperands (7.1,7.2)
   (c) InsufficientPropertyExpressions (i.e. 9.2.3)
   (d) InsufficientObjectExpressionOperands (8.1.1,8.1.2)

[7] `http://www.w3.org/TR/owl2-new-features/#F12:_Punning`

(e) InsufficientAxiomOperands (9.1.2-9.1.4)
(f) EmptyOneOfExpression (7.4,8.1.4)

3. Datatype related syntactic violations
   (a) LexicalNotInLexicalSpace (5.7)
   (b) UseOfIllegalFacetRestriction (7.5)

4 Fixing Violations

4.1 Fixing Declaration Failures

Injecting declarations does not impact the logical meaning of an ontology in most cases. Some profile checkers, such as the one in the OWL API[2] or in the owlapi-tools libraries will sometimes believe that a given violation is a declaration failure for an entity that is part of the reserved vocabulary. Repairing this failure will simply cause another kind violation (vio-res). Some patterns of axioms plus declaration failure have ambiguous legal resolutions for OWL 2 DL. For example,

```
C SubClassOf: (P some D).
```

absent any other information, could be declaring a restriction on an object property (P) or a data property (to an unknown datatype, D). OWL 2 DL forbids punning between classes and datatypes (as well as object- and data properties), so this cannot be resolved by declaration for both interpretations. We claim that it is perfectly sensible to prefer the object property/class interpretation[8]. This resolution (with perhaps a bit of care with names from the XSD namespace) would permit dropping declarations as required syntax (thus eliminating a huge class of "silly" errors plus a source of file size bloat). Supporting *warnings* for such cases would allow users who were failing to declare yet desiring the variant interoperation to catch the situation easily. A slight modification would be to require declarations *only when* the inferred declaration is ambiguous. In general, we think its better to highlight potentially rare situations (such as punning or ambiguous situations) than expect that people will maintain unwieldy declaration sets.

4.2 Fixing Inherent Violations of the DL Profile

Not possible without changing the meaning of the ontology, at least in a general way. However, there are bugs here which are merely nominal. Consider the following example drawn from our corpus:

```
locatedIn Characteristics: Transitive.
Wine SubClassOf: locatedIn min 1 Thing.
```

This nominally violates the restriction on non-simple roles appearing in cardinality restrictions. But it is easy to see that this expression can be replaced by an equivalent, non-violating version:

[8] In absence of any contrary evidence, for example in the ABox

```
locatedIn Characteristics: Transitive.
Wine SubClassOf: locatedIn some Thing.
```

We have not, at this time, developed a repairer for this sort of issue. A proper attempt would not only resolve all such cases (that is, for every violating ontology which has a "straightforward" non-violating equivalent, generate that equivalent), but provide "sensible" approximations for the substantially violating ontologies.

4.3 Fixing misuse of reserved vocabulary

There is no straight forward way to fix violations of this type. One might think of rewriting expressions in such a way that the misused entities are pulled into a different (new) namespace, but this might not be desirable in all cases. For example, an ontology engineering environment might exploit $rdfs : label$ to display class names, as well as potential sub-annotation-properties, e.g. $urn : displayScreenLabel$. Analogously, some tools might also support more than just the OWL 2 datatypes — removing that information would be hugely undesirable.

4.4 Fixing Violations of the OWL Syntax

Generally speaking, there are no direct semantics for an expression that is malformed. It would be, however, impractical to simply discard an entire ontology just because it contains some minor syntactic errors. Non-absolute IRIs can be easily replaced by absolute ones without impairing the logical structure of an ontology. Other violations, such as the datatype related ones, are virtually unfixable without simply dropping expressions or entire axioms. The syntactic violations relating to insufficient operands however are sometimes fixable without affecting the semantics. The three types of axioms related to the InsufficientAxiomOperands violation are DisjointClasses, EquivalentClasses and DisjointUnion. If they contain less than two operands, they can simply be dropped. The same goes for the InsufficientIndividuals (SameIndividual, DifferentIndividuals) and InsufficientPropertyExpressions (EquivalentObjectProperties, DisjointObjectProperties, DisjointDataProperties, EquivalentDataProperties, HasKey, SubPropertyOf(ObjectPropertyChain)). EmptyOneOfExpressions (DataOneOf, ObjectOneOf) and InsufficientDataRangeOperands (DataIntersectionOf, DataUnionOf) violations cannot be easily fixed. OneOf expressions are generally part of larger expressions that would need to be adjusted as well, which is not possible without at least potentially violating the authors modelling intentions. The same goes for the the problem with insufficient data range operands, because we cannot simply replace them with a "general" data range without losing something. For violations of the kind InsufficientObjectExpressionOperands (ObjectUnionOf, ObjectIntersectionOf), we propose a minor rewriting that corresponds to the DL semantics:

For ObjectUnionOf:

1. if expression contains no operands, we rewrite the whole expression to owl:Nothing

2. else if expression contains only owl:Nothing, we rewrite the whole expression to owl:Nothing
3. else if expression contains only one operand c1, we rewrite the whole expression to ObjectUnionOf(c1, owl:Nothing)

For ObjectIntersectionOf:

1. if expression contains no operands, we rewrite the whole expression to owl:Thing
2. else if expression contains only owl:Thing, we rewrite the whole expression to owl:Thing
3. else if expression contains only one operand c1, we rewrite the whole expression to ObjectIntersectionOf(c1,owl:Thing)

4.5 Summary: Which Violations can be safely fixed?

For readability, we list the violations that can be safely fixed[9] without changing the meaning of an ontology.

1. Declaration failures (vio-dec):
   - UseOfUndeclaredClass (inject declaration)
   - UseOfUndeclaredObjectProperty (inject declaration)
   - UseOfUndeclaredDatatype (inject declaration)
   - UseOfUndeclaredDataProperty (inject declaration)
   - UseOfUndeclaredAnnotationProperty (inject declaration)
2. Non-absolute IRIs (vio-syn):
   - UseOfNonAbsoluteIRI (rewrite)
   - OntologyIRINotAbsolute (rewrite)
   - OntologyVersionIRINotAbsolute (rewrite)
3. Some of the insufficient operands violations (vio-syn):
   - InsufficientPropertyExpressions (drop axiom)
   - InsufficientObjectExpressionOperands (rewrite)
   - InsufficientAxiomOperands (drop axiom)
   - InsufficientIndividuals (drop axiom)

5 Materials and Methods

A comprehensive survey of OWL 2 DL profile violations was conducted over MOWLCorp (original serialisations)[5][10], a corpus of 21K ontologies (unique files containing at least a single TBox axiom). A previous version of the corpus was described in detail elsewhere [4].

For violation checking, we used an extended and modified version of the profile checker shipped with the owlapi-tools[11]. The experiment was implemented in Java, building on the OWL API (version 3.5.0)[2].

Note: The *UseOfNonAbsoluteIRI* violation was not fixed for this survey (future work), so it will be listed in a separate category (fixable).

[9] That is, assuming that our ambiguous declaration interpretation is accepted.

[10] Find more information and summary statistics here: `http://mowlrepo.cs.manchester.ac.uk/datasets/mowlcorp/`

[11] `https://github.com/owlcs/owlapitools`

5.1 Data Gathering

For every (parseable) ontology in the corpus we gather its static metrics (axiom counts, entity counts, profile memberships, etc) and its DL profile violations, apply the fixes described in section 4.5, and observe how many ontologies were safely re-writeable to fall under the DL profile without affecting their logical structure.

6 Results

Repair data was successfully gathered for 20,137 ontologies in the corpus (out of 20,995). Some ontologies in the corpus were unloadable due to unavailable imports (695 ontologies), parsing problems[12] (132 ontologies) or other reasons (31 ontologies). As can be seen from Table 1, roughly 70% of all occurring violations can be safely fixed according to our classification scheme without affecting the logical meaning of the ontology (mainly, but not only, due to the fact that missing declarations cause 70% of all violations). Table 2 shows the number of successful and unsuccessful repairs. 8,255 ontologies that did not previously fall under the OWL 2 DL profile were successfully repaired, a large number of which actually turned out to fall under the OWL 2 RL profile (80%). 2,541 ontologies, roughly 13% of the corpus, underwent failed repair attempts. For all of these ontologies, the attempt to inject a missing declaration caused yet another illegal punning, or another "NonAbsoluteIRI" violation, which explains the fact that the total numbers of illegal punning violations (and non absolute IRI ones) actually rose during the repair (Table 3). However, these additional violations were not caused by the repair: They were already there before, and either not correctly detected (incomplete profile checker) or double counted (the profile checker counts violations per axiom, and if 2 axioms cause an illegal punning, the violation is counted twice).

Table 6 shows a detailed breakdown of the violation with the number of ontologies having at least one such violation. IllegalPunning and UnknownDatatype are the two largest classes (and are largely disjoint). UnknownDataype is arguably not an error per se, although if no system can handle that datatype they are functionally so.

7 Discussion

Two striking features of the results stand out: the distribution of ontologies into profiles and the relatively small number of certain sorts of violation e.g., non simple roles in cardinality restrictions. While this is not a profile survey per se, we were surprised by the large number of OWL RL ontologies after repair. Since OWL RL has an OWL Full friendly design, we need to examine the repairs more

[12] The new OBO parser in the latest OWL API version appears to be less forgiving then the old one.

Table 1. The total number of violations before and after repair, by category. For detailed breakdown, see Table 6.

	After	Before	% Repair
Declaration Failure	0	16,993,219	100.00%
Inherent DL	7,329,499	7,277,524	-0.71%
Misused Vocabulary	208,069	208,069	0.00%
Syntactic Violation	91,677	95,051	3.55%
Unfixable	7,538,030	7,486,055	-0.69%
Fixable	0	16,997,550	100.00%
Potentially fixable	91,215	90,258	-1.06%
All	7,629,245	24,573,863	68.95%

Table 2. Profile Membership of parseable files before and after the repair. Notice that two files became OWL 2 ontologies as a result of repair.

		Before	After	Diff
Exclusive	OWL2	16,270	8,025	-8,245
	DL	2,249	3,342	1,093
	EL	180	342	162
	QL	36	45	9
	RL	294	6,178	5,884
OWL EL +	QL	185	508	323
	RL	62	107	45
OWL QL +	RL	43	68	25
OWL EL + QL + RL		525	1,231	706
Total		19,844	19,846	2

closely to see if we are altering potentially significant semantics. It would also be interesting to have a better understanding of what makes them OWL RL (e.g., the gap between these ontologies being in RL and being also in EL). We note that from a constructor perspective, OWL RL allows quite a few (albeit in restricted locations), so that might suffice to explain its large numbers.

The fact that a large number of violating ontologies merely have unknown datatypes is very promising: Adding datatypes to OWL 2 is, by design, rather straightforward. Thus, a potentially easy extension to OWL 2 might bring significant benefit. Alternatively, if most of these uses are mere legacy datatypes, then a tool could potentially significantly help by suggesting modern alternatives.

Illegal punning and reserved vocabulary use are perhaps not as simply amenable to specification tweaks. This awaits further investigation.

Table 3. All violations before (B) and after (A) the repair. $|\mathcal{O}|$ corresponds to the number of affected ontologies by a particular violation. Grouped by bins, sorted by number of affected ontologies before the repair.

Violation	Total (B)	Total (A)	Fixed	$\|\mathcal{O}\|$ (A)	$\|\mathcal{O}\|$ (B)
UndeclaredAnnotationProp	5,704,211	0	100.00%	10,162	0
UndeclaredClass	2,665,962	0	100.00%	9,424	0
UndeclaredObjProp	8,498,762	0	100.00%	2,385	0
UndeclaredDataProp	75,712	0	100.00%	779	0
UndeclaredDatatype	48,572	0	100.00%	245	0
UnknownDatatype	90,247	90,247	0.00%	3,756	3,756
ResVocForClassIRI	60,756	60,756	0.00%	2,081	2,081
ResVocForObjPropIRI	8,548	8,548	0.00%	1,086	1,086
ResVocForIndividualIRI	15,917	15,917	0.00%	774	774
ResVocForDataPropIRI	2,380	2,380	0.00%	548	548
ResVocForAnnotationPropIRI	30,214	30,214	0.00%	321	321
ResVocForOntologyIRI	7	7	0.00%	7	7
ResVocForVersionIRI	0	0	-	0	0
IllegalPunning	7,273,783	7,325,760	-0.71%	4,056	4,056
NSimPropInCardinalityRestriction	2,852	2,852	0.00%	151	151
NSimPropInFunctionalProp	51	51	0.00%	36	36
NSimPropInDisjointProperties	621	621	0.00%	15	15
PropInChainCausesCycle	67	65	2.99%	15	14
NSimPropInAsymmetricObjPropAx	86	86	0.00%	10	10
NSimPropInInverseFunctionalObjProp	9	9	0.00%	9	9
NSimPropInIrreflexiveProp	48	48	0.00%	7	7
TopDataPropAsSubPropInSubPropAx	3	3	0.00%	3	3
CycleInDatatypeDefinition	4	4	0.00%	1	1
BuiltInDatatypeInDatatypeDefinition	0	0	-	0	0
DefinedDatatypeInDatatypeRestriction	0	0	-	0	0
NSimPropInObjectHasSelf	0	0	-	0	0
InsufficientIndividuals	384	0	100.00%	316	0
InsufficientObjectExpressionOperands	2,739	0	100.00%	295	0
NonAbsoluteIRI	90,258	91,215	-1.06%	153	153
LexicalNotInLexicalSpace	462	462	0.00%	136	136
InsufficientAxOperands	1,147	0	100.00%	63	0
InsufficientPropExpressions	60	0	100.00%	40	0
OntologyIRINotAbsolute	1	0	100.00%	1	0
OntologyVersionIRINotAbsolute	0	0	-	0	0
IllegalFacetRestriction	0	0	-	0	0
EmptyOneOfExpression	0	0	-	0	0
InsufficientDataRangeOperands	0	0	-	0	0

One way to conceptualise what we have done in this paper compared with [1] is to update their work on OWL 1 to OWL 2. OWL 2 tried to narrow the gap some between OWL DL and OWL Full, though in also introduced a more complex syntax for OWL DL (to handle new features such as role chains). Furthermore, some features in OWL 2, in particular declarations, introduce new classes of very common violations. In [9], only 18% of the OWL Full documents remained so after patching in comparison to the 47% remaining in our experiments. Its not clear that we can make any useful generalisations from these facts, as the corpora in each paper are very different and neither the repairs nor the detection of violations nor even the base language (OWL 1 vs. OWL 2) are the same. Investigating the residual OWL Full ontologies in our corpus in order to understand how OWL Full is being used is critical future work.

8 Conclusions

There is clearly too much noise generated by the current violation regime. There is no reason to burden users with declaration fiddling and clearly many users and tools simply will not bother. In this sense, the market seems to have spoken. An interactive approach, where users could be made aware of particular violations and their consequences and then are suggested ways to repair them, might be an interesting add-on for existing debugging tools. Overall, we are encouraged that it will be possible to significantly close the gap between arbitrary OWL 2 and OWL 2 DL ontologies.

References

1. S. Bechhofer and R. Volz. Patching Syntax in OWL Ontologies. In *International Semantic Web Conference*, pages 668–682, 2004.
2. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2:11–21, 2011.
3. I. Horrocks and U. Sattler. A tableau decision procedure for *shoiq*. *J. Autom. Reasoning*, 39(3):249–276, 2007.
4. N. Matentzoglu, S. Bail, and B. Parsia. A Snapshot of the OWL Web. In *International Semantic Web Conference (1)*, pages 331–346, 2013.
5. N. Matentzoglu and B. Parsia. The Manchester OWL Corpus (MOWLCorp), original serialisation. July 2014.
6. B. Motik, B. Parsia, and P. Patel-Schneider. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), 2012.
7. P. F. Patel-Schneider. Building the semantic web tower from rdf straw. In *IJCAI*, pages 546–551, 2005.
8. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5:51–53, 2007.
9. T. D. Wang, B. Parsia, and J. A. Hendler. A Survey of the Web Ontology Landscape. In *International Semantic Web Conference*, pages 682–694, 2006.