$\text{OWL}_{\text{strict}}$: A Constrained OWL Fragment to avoid Ambiguities for Knowledge Graph Practitioners

Robert David[1,2][0000−0002−3244−5341], Albin Ahmeti[1,3][0000−0001−8766−4069], Shqiponja Ahmetaj[3][0000−0003−3165−3568], and Axel Polleres[2,4][0000−0001−5670−1146]

[1] Semantic Web Company
[2] Vienna University of Economics and Business (WU Wien)
[3] Vienna University of Technology (TU Wien)
[4] Complexity Science Hub Vienna

Abstract. While RDF is a standard exchange format for directed labelled graphs, the Web Ontology Language (OWL) is a likewise standardized, powerful and flexible language to express terminological axioms for describing the schema of RDF graphs. As such, RDF and OWL are natural choices for interchanging and describing the schema of Knowledge Graphs (KGs) in practice. However, the syntactic "freedom" of RDF can lead to unintuitive OWL ontologies that are challenging for implementers of KGs to use and which can cause ambiguous interpretations when used in ontology editors, or – even worse – hint at modeling errors. In this paper, starting from the notion of "standard-use" of RDF, we aim at providing a methodology and tool support for the challenge of solving respective ambiguity problems in the context of ontology editors. We first define a syntactically constrained fragment of OWL, $\text{OWL}_{\text{strict}}$, which aims to avoid ambiguities based on requirements identified in ontology editors in practice. To strengthen the argument for avoiding these ambiguities, we compare how different ontology editors handle them. Second, we discuss how far this fragment can be syntactically described and checked using SHACL. Third, to provide tool support and building upon our prior work on SHACL repairs, we discuss how non-$\text{OWL}_{\text{strict}}$-compliant ontologies can be repaired, i.e., modified to be made compliant with $\text{OWL}_{\text{strict}}$. We perform an evaluation with a particular ontology editor and six ontologies we use in real-world projects and show that the repaired versions satisfy $\text{OWL}_{\text{strict}}$ conformance requirements and are therefore compatible with the editor.

Keywords: OWL, Web Ontology Language, Ontology editor, knowledge graphs, constraints, SHACL, Shapes Constraint Language, data repairs.

1 Motivation

The Web Ontology Language (OWL) [3] is the standard language of the Semantic Web to express ontologies as a way to describe schemas for KGs. However,

the flexibility of representing graph data using RDF triples, which itself does not restrict the use of classes and properties on the syntactic level, might lead to ambiguous OWL schemas, which makes it difficult or even problematic in use cases requiring high quality schema descriptions in practice. Two of such closely related use cases are data integration and applications built on top of KG data. Originally designed to be used on the (public) Web, Semantic Web technologies were discovered to be sufficiently flexible and expressive to be used in enterprise contexts to solve data integration challenges under the term Enterprise Knowledge Graphs (EKG) [10]. In EKG scenarios, besides having the data exposed and accessible using standards like RDF and SPARQL, applications and tools processing EKG data are likewise important.

While the vast majority of enterprise ontologies are being created by knowledge engineers using ontology editors, the syntactic flexibility of representing OWL as (asserted) RDF triples imposes challenges on such editors, especially when it comes to importing existing published ontologies. Sufficient information might be missing for tools to interpret ontology entities without ambiguities, or there might even be contradictory information, which is not necessarily understood and detected by OWL reasoners, because axioms are missing to derive these contradictions. Generally, OWL is not designed for syntax conformance[5]. Syntactic restrictions cannot be enforced, which consequentially might lead to ambiguous and unintuitive interpretations. To avoid such situations, there is a need for clearly defined schemas, which are syntactically well-defined and unambiguous. "Well-defined" in our context means there is sufficient information regarding OWL definitions in the schema so they can be understood and classified by tools without ambiguities, e.g., every node in the ontology has a clear and distinct class membership regarding built-in OWL classes and properties.

In this paper we describe specific common challenges identified in a number of published ontologies we studied and which can be problematic for ontology editors. We particularly look at editors applied in real-world scenarios that are susceptible to such ambiguities, consequently causing them to not work correctly or provide misleading information. We identify requirements for restrictions and suggest $\text{OWL}_{\text{strict}}$ as a restricted but practically useful OWL fragment, that – rather than driven by decidability and complexity of reasoning – is driven by practical applicability in ontology editors. $\text{OWL}_{\text{strict}}$ defines a set of constraints within a syntactic fragment of OWL. Unlike standard OWL, which operates under the open world assumption (OWA), $\text{OWL}_{\text{strict}}$ enforces these constraints to ensure that the ontology RDF graph syntactically adheres to them. For evaluation purposes, we address these challenges from the viewpoint of a particular ontology editor, PoolParty, for which the identified restrictions provide a solution to avoid ambiguities.

In the following, we discuss a motivating example from the published DITA [8] ontology for describing documents.

[5] `https://www.w3.org/TR/owl2-primer/`

Example 1. An example of an ambiguous situation in DITA is the incoherent use of classes and datatypes in declared domains and ranges of ObjectProperties or DatatypeProperties.

```
:referenceObject a owl:ObjectProperty ;
    rdfs:subPropertyOf :element ;
    rdfs:domain :Doctype ;
    rdfs:range :ReferenceObject .

:category  a rdf:Property ;
    rdfs:subPropertyOf :element ;
    rdfs:range xsd:string .

:element a rdf:Property ;
    rdfs:domain :Doctype ;
    rdfs:range rdfs:Resource .

:Doctype a rdfs:Class .
:ReferenceObject a rdfs:Class.
```

This example shows an ambiguity regarding the property *element*, which has two subproperties *referenceObject* and *category*. *referenceObject* is asserted to be a member of ObjectProperty. *category* is asserting a range of xsd:string, which implies membership in DatatypeProperty. *element*, however, does not assert a membership in either ObjectProperty or DatatypeProperty, which leaves tools which need such a classification with the challenge of interpretation.

element, having both an ObjectProperty and a DatatypeProperty as subproperties, can be interpreted as a member of either one of these disjoint classes. An OWL reasoner, interpreting *element* as an ObjectProperty based on the OWL semantics [25], would not infer an *element* property for an asserted *category* property because of the disjointness of classes, which unintuitively ignores this asserted subproperty relation. An intuitive and consistent class membership for *element* in one of the classes ObjectProperty or DatatypeProperty is an open challenge. However, such a classification is a common requirement in ontology editors.

Our contributions are the following:

- We investigate a number of ontologies used in real-world projects and tools for ontology editing, from which we identify requirements to avoid ambiguities.
- We define the OWL$_{\text{strict}}$ fragment, which aims at avoiding such ambiguities and unintuitive descriptions by requiring ontology entities to be well-defined and by establishing syntactic restrictions on ontology RDF graphs.
- We show how checking OWL$_{\text{strict}}$ fragment conformance can be implemented using the Shapes Constraint Language (SHACL) [19].
- Building upon recent work for SHACL repairs[6] [1], we provide tool support to transform an OWL ontology RDF graph into an OWL$_{\text{strict}}$ conformant

[6] `https://github.com/robert-david/shacl-repairs`

RDF graph with a notion of only minimally changing it syntactically (but not semantically).

Our work is structured as follows: we present related work on existing OWL fragments, syntactic restrictions, standard-use of OWL and RDFS, and ontology repairs in Section 2. We follow with a discussion of ontology editors and how ambiguities are addressed and define $\text{OWL}_{\text{strict}}$ to avoid these ambiguities in Section 3. We then present SHACL constraints which verify $\text{OWL}_{\text{strict}}$ conformance in Section 4. We continue with the evaluation in Section 5 and apply SHACL repairs on six ontologies for various domains used in real-world projects with the PoolParty ontology editor. These six ontologies are modified for $\text{OWL}_{\text{strict}}$ conformance and are consequently verified to be compatible with the editor, thereby making them available for consistent use by users. Finally we conclude by discussing future work for $\text{OWL}_{\text{strict}}$ in Section 6.

2 Related Work

Related to our work are the "official" OWL 2 profiles[7] published as W3C recommendations. Other OWL fragments, like the previous version OWL 1 and unofficial fragments introduced for specific use cases, are also presented. Besides OWL fragments, we show other related work about intuitive syntactic restrictions and standard-use on OWL and RDFS to avoid unintended use. Finally, different approaches for repairing ontologies are presented.

Existing OWL Fragments RDF, RDFS and OWL are defined as W3C recommendations and have seen development over multiple versions. The current version OWL 2 is based on Description Logics (DL) and defines fragments as use case profiles: EL for large numbers of classes and properties, QL for query rewriting into SQL, and RL to express rules and scalable reasoning. OWL 2 profiles are defined by placing syntactic restrictions on the structure of ontologies to trade some expressive power for efficiency of reasoning. Besides fragments defined in W3C recommendation, various non-standard fragments were published to address further use cases. Some work, like [22], even address interpretations as constraints for scenarios where data integrity is important. Table 1 gives an overview of the existing RDFS and OWL fragments with brief descriptions.

We note that these published OWL (and RDFS) fragments do not necessarily address our scenario to avoid ambiguities at the syntactic level. Some work, like [23], is closely related to ours by restricting to a small fragment for practical use, while others, like [26] and [9], introduce constraints as restrictions on the data. However, none of the listed fragments fully addresses our problem setting.

Syntactic restrictions on OWL and RDFS vocabulary usage While not defined as "fragments", several intuitive syntactic restrictions on the usage of OWL and RDFS have been defined in the literature, which are motivated by

[7] `https://www.w3.org/TR/owl2-profiles/`

Name	W3C Rec.	Purpose
RDF 1999	yes	first RDF recommendation version
RDF 1.0	yes	February 2004
RDF 1.1	yes	June 2014
RDFS 1.0	yes	schema description
RDFS 1.1	yes	schema description
Minimal RDF(S) [23]	no	small and practical
OWL 1 Lite	yes	minimum complexity
OWL 1 DL	yes	Description logics
OWL 1 Full	yes	maximum expressivity
OWL 2 EL	yes	many classes & properties
OWL 2 QL	yes	query rewriting
OWL 2 RL	yes	rules & reasoning
RDFS++	no	AllegroGraph's minimal OWL profile
OWL Horst [17]	no	scalable reasoning
OWLSIF	no	Oracle's OWL with if-semantics
OWLPrime	no	Oracle's application-oriented OWL profile
OWL DBP [2]	no	DBPedia schema
OWL Wikidata	no	Wikidata schema
OWL LD [11]	no	Linked data practical use
OWL IC [26]	no	integrity constraints
OWL Flight [9]	no	logic programming & constraints

Table 1. OWL Fragments.

avoiding "abuse" of RDF not restricting how classes and properties are used. The "Pedantic Web group"[8] started to collect such arguable abuse problems in 2009, including the following issues [15]:

- syntactic problems that yield a whole RDF document may be obvious and related to a particular serialization syntax such as non-closed tags in RDF/XML, or missing '.'s in Turtle to conclude triple groups, but may also be more subtle, including syntactically non-allowed IRIs references containing non-IRI-encoded special characters.
- unintended use of the RDF container, collection and reification vocabularies, such as - for blank nodes in the object of an `rdf:predicate` triple, or structurally wrong usage of the RDF list vocabulary (cyclic, non-closed or non-linear lists).
- unintended "meta-modelling", such as using classes as properties and vice versa, or using classes/properties as instances.
- literals incompatible with datatypes: a datatype in RDF in D-Entailment should come along with a syntactic definition of valid lexical forms for liter-

[8] https://web.archive.org/web/20091005212138/http://pedantic-web.org/

als within this datatype, often/typically defined and therefore syntactically checkable by e.g., a regular expression.
– incoherent (i.e., wrong and mixed) use of literals and IRIs in declared ObjectProperties or DatatypeProperties.

Apart from these, Hogan et al. [15] also lists other suspicious issues, that are not necessarily syntactically traceable, but can also be problematic. Partially, checks for the prevalence of these issues have been implemented, but not formally defined, in the – in the meantime discontinued[9] – RDFAlerts service.

Standard RDF(S) and OWL vocabulary use The notion of standard-use of the reserved RDF, RDFS and OWL vocabularies, identified by the respective namespaces intuitively, refers to the idea that the respective classes and properties defined by these vocabularies should not be syntactically abused or re-defined. Without reiterating respective formal definitions from the literature (Def. 5.5 in [14], or respectively Def. 3.3 in [13]), essentially *standard use* restricts the usage of the reserved RDF, RDFS and OWL vocabularies as follows.

Definition 1. *Let $C_{res} = C_{RDF} \cup C_{RDFS} \cup C_{OWL}$ be the reserved classes in the RDF, RDFS and OWL vocabularies, and analogously $P_{res} = P_{RDF} \cup P_{RDFS} \cup P_{OWL}$ be the resp. properties in these vocabularies. Then*

– *elements from C_{res} may only appear in object positions of* `rdf:type` *triples.*
– *elements from P_{res} may only appear in predicate positions.*

This condition can be easily checked by the following SPARQL pattern:

```
{ ?s ?p ?o .
FILTER(?s NOT IN P_res)   FILTER(?o NOT IN P_res)
FILTER(?s NOT IN C_res)   FILTER(?p NOT IN C_res)
FILTER(?p = rdf:type || ?o NOT IN C_res) }
```

In addition to disallowing any other terms than P_{res} and C_{res} using the RDF, RDFS and OWL namespaces, we may define standard-use as a purely syntactic filter on the terms appearing in triples. Note that the notion of "standard use" could be tightened in various respects, avoiding different situations that potentially are hard to handle for tools:

– disallowing non-standard Datatypes (beyond XSD-Datatypes mentioned in the RDF Standard) in the context of D-Entailment, or at least checking the syntactical validity of known XSD Datatypes.
– disallowing literals in the range of properties used with different lexically disjoint datatypes (applying similar considerations as for object property ranges).
– requiring disjointness of Classes and Datatypes, or checking compatibility of range classes with datatypes.
– disallowing other meta-modelling facilities such as usage of classes as properties, classes as instances, or enforcing "punning" (à la OWL 2).

[9] https://github.com/timrdf/DataFAQs/wiki/RDFAlerts

Repairing Ontologies Closely related to our work are reasoning tasks for repairing ontologies with different kinds of definitions of what a repair means.

Debugging and repair of OWL ontologies [18] addresses semantic defects in OWL-DL ontologies. Inconsistencies to be repaired are logical contradictions that manifest as either inconsistent ontologies or unsatisfiable concepts. It focuses on rewriting the ontology axioms to mitigate the semantic defects based on multiple approaches, including a relevance scoring, as opposed to our work, which focuses on syntactic disambiguities affecting tools.

A recent work on Description Logic ontologies is [4], which addresses optimal repairs in the scenario where the schema (TBox) is assumed to be correct, while the data (ABox) needs to be repaired. Optimal repairs in this context are repairs that preserve as much as possible from the logical consequences of the ontology, which differs from our work, which modifies the TBox to avoid syntactic disambiguities independent from ABoxes.

An approach to repair EL ontologies is presented in [20], which focuses on the TBox itself and does not consider the ABox. It uses a combination (or trade-off) of weakening and completing to mitigate the negative effects of removing wrong axioms in the ontology. This work with a focus on the TBox is closest to ours. However, we focus on ambiguities with axioms only and do not take semantic consequences into account.

3 OWL$_{\text{strict}}$

We first present several tools for ontology editing, discuss how they address ambiguities and identify requirements to keep ontologies unambiguous. We then define OWL$_{\text{strict}}$ as an OWL fragment, including a definition of "well-defined" and syntactic restrictions, to implement these requirements.

3.1 Ontology Tools and how they address Ambiguities

Protégé Protégé [10] is an OWL ontology editor which allows a lot of flexibility regarding imported ontology RDF triples. Protégé does not restrict ontologies besides the RDF data model. It will interpret the imported RDF triples for presenting OWL entities in the user interface. For example, having an RDF triple with a predicate of *rdfs:subPropertyOf* and no other information about the subject or object, Protégé will interpret them as owl:AnnotationProperty members, whereas by adding another RDF triple which defines a *rdfs:range* using the same subject and a resource as the object, it will be interpreted as an owl:ObjectProperty. Generally, Protégé allows ambiguities in ontology editing and does not support resolving them.

TopBraid Composer Similar to Protégé , TopBraid Composer[11] provides an ontology editor that provides a lot of flexibility and supports the syntax of all

[10] `https://protege.stanford.edu/`
[11] `https://topbraidcomposer.org/html/`

elements of OWL 2. The editor avoids ambiguities by providing generic views, and it does not interpret imported RDF triples. TopBraid Composer supports SPARQL Inference Notation (SPIN)[12] and SHACL, which can both be used for user-defined consistency checks.

PoolParty Semantic Suite PoolParty Semantic Suite[13] is a commercial product, which provides an ontology editor as a core component. PoolParty supports a specific non-standard OWL fragment intended for EKG modeling, which was developed based on real-world requirements in projects, and which requires clear and unambiguous schemas. Complete (open world) OWL reasoning is generally avoided in typical use cases, because they rather center around data consistency and quality control, where the OWA has limited use. Ontologies need to conform to certain syntactic restrictions such that ambiguities are avoided and entities are clearly defined as a prerequisite to support data quality. While the ontology editor prevents ambiguities for ontologies modeled in the tool itself, this is not necessarily the case for ontologies which are imported into the system for reuse, which is a common case. For evaluation purposes, we choose PoolParty as a representative editor because of strict restriction requirements originating in real-world scenarios.

Further ontology editing tools Before the introduction of SHACL as a constraint standard to check conformance for KGs, various vendors adopted the idea of interpreting ontologies as closed-world in the sense of interpreting OWL elements similar to SHACL constraints for integrity constraint checks. Triple store vendors, e.g., Stardog, defined constraint semantics for OWL axioms, the so-called Stardog Integrity Constraint Validation (ICV)[14]. Another example is Metaphactory [12], which translates OWL ontologies into SHACL constraints during the import phase of an ontology, and afterwards uses these constraints for integrity checking.

3.2 Identified Requirements

We identify specific requirements to avoid ambiguities, motivated by strict real-world requirements of the PoolParty editor, as:

R-1 All ontology entities defined in the tool have an asserted membership in exactly one of the OWL built-in classes OWLClass, Datatype, ObjectProperty and DatatypeProperty (without running an OWL reasoner), so that the user interface can provide a clear separation based on these classes for users to model entities.

R-2 Consistent OWL built-in class membership with respect to property hierarchies, e.g., a subproperty of an object property is also an object property.

[12] `http://spinrdf.org/`

[13] `http://poolparty.biz`

[14] `https://docs.stardog.com/data-quality-constraints`

R-3 Domain and range definitions are limited to a maximum of one domain and range triple per property. We want to avoid intersection of different classes when ontologies define several domain and range axioms, which many users in practice find unintuitive, but support a (choice) set of classes, which is represented as an OWL union (owl:unionOf).

R-4 Consistency for domain and range classes with respect to property hierarchies: the domain class of a subproperty needs to be the same class or a subclass of a (one of the) domain class(es) of the superproperty, and the same is true for range classes and datatypes.

3.3 OWL$_{\text{strict}}$ Fragment

OWL$_{\text{strict}}$ represents an approach to clearly describe nodes in an RDF graph so that they can be understood without ambiguities regarding the OWL$_{\text{strict}}$ vocabulary, thereby conforming to OWL$_{\text{strict}}$. We define this unambiguity only for nodes which occur in atoms using the OWL$_{\text{strict}}$ vocabulary. In the following, we first define the OWL$_{\text{strict}}$ vocabulary, where we identified the main issues regarding ambiguities, followed by the conditions of well-definedness and syntactic restrictions, which an RDF graph needs to conform to in order to qualify as an OWL$_{\text{strict}}$ ontology.

Definition 2. *Assume the finite disjoint sets C and P of* OWL$_{\text{strict}}$ *built-in class names and property names, respectively, which form the* OWL$_{\text{strict}}$ *vocabulary, and the countably infinite set of* nodes N *disjoint with C and P. We define C and P as follows.*

$$\begin{aligned} C = \{&\mathsf{RDFSClass}, \mathsf{OWLClass}, \mathsf{Datatype}, \mathsf{Property}, \mathsf{ObjectProperty}, \\ &\mathsf{DatatypeProperty}, \mathsf{AnnotationProperty}, \mathsf{OntologyProperty}, \\ &\mathsf{SymmetricProperty}, \mathsf{FunctionalProperty}\} \\ P = \{&\mathit{subClassOf}, \mathit{subPropertyOf}, \mathit{domain}, \mathit{range}, \\ &\mathit{inverseOf}, \mathit{disjointWith}\} \end{aligned}$$

Definition 3. *An* OWL$_{\text{strict}}$ *RDF graph G is a finite set of atoms of the form $B(a)$ and $p(b, c)$, where $B \in C$, $p \in P$ and $a, b, c \in N$. The (finite) set of nodes appearing in G is denoted with $V(G)$.*

A node $n \in V(G)$ is said to be well-defined in G if $B(n) \in G$ and $B, B' \in \{\mathsf{OWLClass}, \mathsf{Datatype}, \mathsf{ObjectProperty}, \mathsf{DatatypeProperty}\}$ and there is no $B'(n) \in G$ such that $B \neq B'$.

G is said to be well-defined if every node in $V(G)$ is well-defined.

Well-defined in the OWL$_{\text{strict}}$ context means there is sufficient axiomatic description for nodes in $V(G)$ so they can be understood without ambiguities, i.e., every node in $V(G)$ has a unique class membership in G regarding the four built-in OWL$_{\text{strict}}$ classes OWLClass, Datatype, ObjectProperty and DatatypeProperty.

Similar to OWL 2 profiles, an OWL$_{\text{strict}}$ RDF graph G follows syntactic restrictions on atoms in G depending on the specific names in C and P. We

define the syntactic restrictions in the spirit of global restrictions for OWL 2 DL ontologies[15].

Definition 4. *Let G be an* $\text{OWL}_{\text{strict}}$ *RDF graph. G conforms to* $\text{OWL}_{\text{strict}}$ *if atoms in G obey the syntactic restrictions SR-0 to SR-25.*

In the following, we define the syntactic restrictions on G and refer to the requirement in section 3.2 that they implement. We note that some, but not all, restrictions are addressed by OWL semantics as well.

Syntactic restrictions on OWL class membership (R-1)

SR-0 There is no node in $V(G)$ that belongs to more than 1 of the classes OWLClass, Datatype, ObjectProperty and DatatypeProperty.

Syntactic restrictions on classes (R-1)

SR-1 If $\mathsf{RDFSClass}(x) \in G$, then $\mathsf{OWLClass}(x) \in G$.
SR-2 If $\mathit{subClassOf}(x, y) \in G$, then $\mathsf{OWLClass}(x), \mathsf{OWLClass}(y) \in G$.
SR-3 If $\mathit{disjointWith}(x, y) \in G$, then $\mathsf{OWLClass}(x), \mathsf{OWLClass}(y) \in G$.

Syntactic restrictions on properties (R-1)

SR-4 If $\mathit{subPropertyOf}(x, y) \in G$, then x and y are both members of exactly one class of ObjectProperty or DatatypeProperty.
SR-5 If $\mathit{domain}(x, y) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.
SR-6 If $\mathit{range}(x, y) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.
SR-7 If $\mathsf{FunctionalProperty}(x) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.
SR-8 If $\mathsf{SymmetricProperty}(x) \in G$, then $\mathsf{ObjectProperty}(x) \in G$.
SR-9 For each x, y in $\mathit{inverseOf}(x, y) \in G$, $\mathsf{ObjectProperty}(x), \mathsf{ObjectProperty}(y) \in G$.
SR-10 If $\mathsf{Property}(x) \in G$, then x is a member of exactly one class of ObjectProperty or DatatypeProperty.

Syntactic restrictions on domain and range (R-1)

SR-11 If $\mathit{domain}(x, y) \in G$, then $\mathsf{OWLClass}(y) \in G$.
SR-12 If $\mathit{range}(x, y), \mathsf{ObjectProperty}(x) \in G$, then $\mathsf{OWLClass}(y) \in G$.
SR-13 If $\mathit{range}(x, y), \mathsf{DatatypeProperty}(x) \in G$, then $\mathsf{Datatype}(y) \in G$.
SR-14 If $\mathit{domain}(x, y), \mathsf{SymmetricProperty}(x) \in G$, then $\mathit{range}(x, y) \in G$.
SR-15 If $\mathit{range}(x, y), \mathsf{SymmetricProperty}(x) \in G$, then $\mathit{domain}(x, y) \in G$.
SR-16 If $\mathit{domain}(x, z), \mathit{inverseOf}(x, y) \in G$, then $\mathit{range}(y, z) \in G$.
SR-17 If $\mathit{range}(x, z), \mathit{inverseOf}(x, y) \in G$, then $\mathit{domain}(y, z) \in G$.

[15] `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Global_Restrictions_on_Axioms_in_OWL_2_DL`

Syntactic restrictions on the property hierarchy (R-2)

SR-18 If $subPropertyOf(x, y), \mathsf{ObjectProperty}(x) \in G$, then $\mathsf{ObjectProperty}(y) \in G$.
SR-19 If $subPropertyOf(x, y), \mathsf{DatatypeProperty}(x) \in G$, then $\mathsf{DatatypeProperty}(y) \in G$.
SR-20 If $subPropertyOf(x, y), \mathsf{FunctionalProperty}(y) \in G$, then $\mathsf{FunctionalProperty}(x) \in G$.
SR-21 If $subPropertyOf(x, y), \mathsf{SymmetricProperty}(y) \in G$, then $\mathsf{SymmetricProperty}(x) \in G$.

Cardinality restrictions on domain and range (R-3)

SR-22 If $domain(x, y1), domain(x, y2) \in G$, then $y1 = y2$.
SR-23 If $range(x, y1), range(x, y2) \in G$, then $y1 = y2$.

Syntactic restrictions on domain and range in the property hierarchy (R-4)

SR-24 If $subPropertyOf(x, y), domain(y, z1) \in G$, then $domain(x, z2) \in G$ and either $z1 = z2$ or $subClassOf(z2, z1)$.
SR-25 If $subPropertyOf(x, y), range(y, z1) \in G$, then $range(x, z2) \in G$ and either $z1 = z2$ or $subClassOf(z2, z1)$.

These restrictions establish explicit descriptions in the OWL$_{\text{strict}}$ RDF graph. They ensure that ambiguous node descriptions are avoided regarding the OWL$_{\text{strict}}$ vocabulary, i.e., every node in an OWL$_{\text{strict}}$ RDF graph G can be understood without ambiguities.

Definition 5. *An* OWL$_{\text{strict}}$ *ontology RDF graph* G_O *is an* OWL$_{\text{strict}}$ *RDF graph* G*, which is well-defined and conforms to* OWL$_{\text{strict}}$*, i.e., conforms to the syntactic restrictions SR-0 to SR-25.*

In the following, we revisit Example 1 to illustrate how the syntactic restrictions ensure that every node has an unambiguous description.

Example 2. Assume the following OWL$_{\text{strict}}$ RDF graph G.

$$\begin{aligned} G = \{&\mathsf{ObjectProperty}(referenceObject), domain(referenceObject, DocType),\\ &range(referenceObject, ReferenceObject),\\ &subPropertyOf(referenceObject, element),\\ &\mathsf{Property}(category), range(category, XSDString),\\ &subPropertyOf(category, element),\\ &\mathsf{Property}(element), domain(element, DocType), range(element, RDFResource),\\ &\mathsf{RDFSClass}(DocType), \mathsf{RDFSClass}(ReferenceObject), \mathsf{Datatype}(XSDString)\} \end{aligned}$$

G does not satisfy the syntactic restrictions of OWL$_{\text{strict}}$. It violates SR-10, because *category* and *element* are not members of either ObjectProperty or DatatypeProperty. It violates SR-25, because the range definitions are not consistent between the properties in the subproperty relations, where the ranges of *referenceObject* and *category* need to be subsets of the range of *element*. Satisfying SR-10 would lead to SR-0 being violated, because ReferenceObject and XSDString need to be disjoint. Finally, ReferenceObject needs to be an OWLClass and XSDString needs to be a Datatype because of SR-12 and SR-13.

4 Expressing $\text{OWL}_{\text{strict}}$ in SHACL

Similar to checking fragment containment for OWL, like OWL API [16] does, we determine SHACL shapes which capture the $\text{OWL}_{\text{strict}}$ syntactic restrictions. While some restrictions, like disjointness of classes (e.g., SR-0), can be expressed by OWL axioms, the OWL semantics cannot capture all restrictions necessary to fulfill our requirements. Therefore, in the spirit of model checking [5][6], we use SHACL to implement the syntactic restrictions.

4.1 $\text{OWL}_{\text{strict}}$ SHACL Shapes

We capture the syntactic restrictions using SHACL shapes to validate if a given RDF graph contains an $\text{OWL}_{\text{strict}}$ conformant ontology RDF graph. We present RDF for ClassShape, ObjectPropertyShape and DomainConsistencyShape and omit RDF for other shapes for brevity. The complete set of SHACL shapes is published online[16].

The *ClassShape* and the *DatatypeShape* capture members of owl:Class (SR-1, SR-2, SR-3) and rdfs:Datatype, respectively. The two shapes are defined as mutually exclusive (SR-0).

```
:ClassShape a sh:NodeShape;
    sh:targetClass rdfs:Class, owl:Class;
    sh:targetSubjectsOf rdfs:subClassOf, owl:disjointWith;
    sh:targetObjectsOf rdfs:subClassOf, owl:disjointWith;
    sh:class owl:Class; sh:not [ sh:class rdfs:Datatype; ].
```

The PropertyShape captures all properties of an ontology based on class membership (SR-7, SR-8) and use of properties (SR-4, SR-5, SR-6, SR-9). This shape basically makes the decision between the membership with owl:ObjectProperty or owl:DatatypeProperty via disjunction (SR-10). It also limits domain and range to one each (SR-22, SR-23) and defines the domain of all properties to satisfy ClassShape (SR-11). Finally, it also captures the disjointness of properties with classes and datatypes (SR-0).

The ObjectPropertyShape captures members of owl:ObjectProperty and the requirement of owl:ObjectProperty to have members of owl:Class as range (SR-12). It also makes them disjoint with members of owl:DatatypeProperty (SR-0). Similar, DatatypePropertyShape captures members of owl:DatatypeProperty, captures the range requirement of rdfs:Datatype members (SR-13) and makes them disjoint with owl:ObjectProperty (SR-0). Finally, both shapes capture the consistency requirement of class membership with subproperties (SR-18, SR-19).

```
:ObjectPropertyShape a sh:NodeShape;
    sh:class owl:ObjectProperty;
    sh:property [
        sh:path rdfs:range; sh:node :ClassShape; ];
```

[16] https://github.com/robert-david/OWL-strict/blob/main/ontology_repair_shapes.ttl

```
sh:property [
    sh:path ( rdfs:range owl:unionOf
              [ sh:zeroOrMorePath rdf:rest ] rdf:first );
    sh:node :ClassShape; ];
sh:property [
    sh:path rdfs:subPropertyOf;
    sh:class owl:ObjectProperty; ];
    sh:not [ sh:class owl:DatatypeProperty; ].
```

The FunctionalPropertyShape requires a subproperty of an owl:FunctionalProperty to also be a member of owl:FunctionalProperty (SR-20).

The SymmetricPropertyShape captures the implicit requirement of owl:ObjectProperty (SR-8) membership because of the equal domain and range given by the symmetry (SR-14, SR-15). Also, we require a subproperty of an owl:SymmetricProperty to also be a member of owl:SymmetricProperty (SR-21).

Similar to the SymmetricPropertyShape, the InversePropertyShape captures the owl:ObjectProperty membership (SR-9) and the inverse equality of domain and range (SR-16, SR-17).

DomainConsistencyShape and RangeConsistencyShape capture the domain and range consistency requirements of properties with *rdfs:subPropertyOf* relations. Either the superproperty does not define a domain or range, which lets subproperties define their own domain and range, or it defines a domain or range and therefore requires subproperties to define their domain and ranges accordingly (SR-24, SR-25).

```
:DomainConsistencyShape a sh:NodeShape;
    sh:targetSubjectsOf rdfs:subPropertyOf;
    sh:or (
        [ sh:property [
            sh:path ( rdfs:subPropertyOf rdfs:domain );
            sh:maxCount 0; ] ]
        [ sh:property [
            sh:path ( rdfs:subPropertyOf rdfs:domain);
            sh:minCount 1; sh:equals rdfs:domain; ] ] ).
```

4.2 Limitations of SHACL Core

The shapes DomainConsistencyShape and RangeConsistencyShape have limitations regarding capturing some cases of SR-24 and SR-25, which stem from SHACL Core expressivity. If the domain or range is not a single class or datatype, but a set of classes or datatypes represented with *owl:unionOf*, then a subset relation between domains or ranges of subproperties cannot be verified using SHACL Core. However, we can express these cases using SHACL-SPARQL. SubsetDomainShape and SubsetRangeShape capture the cases of SR-24 and SR-25, respectively, where two properties related with *rdfs:subPropertyOf* have a set of classes each. We note that also the previously defined SHACL Core shapes can be expressed using SPARQL [7].

5 Repairing non-OWL$_{\text{strict}}$ Ontologies

In the following, we apply our previous work on SHACL repairs to modify existing ontologies to conform to OWL$_{\text{strict}}$. The SHACL repair program generates modifications as change sets for G, adding atoms A and deleting atoms D. This enables us to (re-)use OWL$_{\text{strict}}$ versions of any published OWL ontology.

Example 3. We revisit Example 2 to illustrate how SHACL repairs can modify O to satisfy the syntactic restrictions.

$$A = \{\mathsf{ObjectProperty}(element), \mathsf{DatatypeProperty}(category),$$
$$\mathsf{OWLClass}(Doctype), \mathsf{OWLClass}(ReferenceObject)\}$$
$$D = \{range(element, Resource), subPropertyOf(category, element)\}$$

We delete the subproperty relations between *category* and *element* and making *category* a DatatypeProperty and *element* an ObjectProperty because of the respective range assertions. We delete the range of *element* to make it consistent with the range of *referenceObject* (by asserting no domain). Also, OWLClass is added for RDFSClass members.

5.1 Evaluation with real-world Scenarios

We performed an evaluation with six published ontologies from different domains which we use in real-world customer projects.

- **Darwin Information Typing Architecture (DITA)** [8] is an XML-based, end-to-end architecture for authoring, producing, and delivering technical information. The DITA RDF project[17] develops an ontology to describe DITA XML objects using Semantic Web standards.
- **gist** [21] is a minimalist upper ontology aimed at organisations to model an ontology for their business. It is intended to be used enterprise-wide or for major sub-areas within large enterprises. gist provides a core module[18], which covers concepts common to most enterprises.
- **European Union Agency for Railways (ERA)**[19] [24] describes the concepts and relationships related to the European railway infrastructure and the vehicles authorized to operate over it.
- **SemOpenAlex**[20], a KG about scientific publications, which is based on OpenAlex[21]. OpenAlex describes scholarly entities and how those entities are connected to each other.
- **EBUCorePlus**[22] is the follow-up ontology on EBUCore[23] and CCDM (Class Conceptual Data Model). It describes audio, video and other resources for broadcasting applications in the context of a Service Oriented Architecture.

[17] `http://purl.org/dita/ditardf-project`
[18] `https://github.com/semanticarts/gist`
[19] `https://data-interop.era.europa.eu/era-vocabulary/`
[20] `https://github.com/metaphacts/semopenalex/`
[21] `https://docs.openalex.org/`
[22] `https://tech.ebu.ch/metadata/ebucoreplus`
[23] `https://tech-metadata.ebu-it-tools.ch/ontologies/ebucore/index.html`

– **Data Product Model (DPROD)**[24] describes and manages data products consistently across platforms and provides standardization needed to ensure interoperability.

First, we applied the SHACL repair program to the original ontologies to automatically convert them into OWL$_{\text{strict}}$ versions. Second, we verified the OWL$_{\text{strict}}$ versions of the ontologies by running the repair program again to see that there are no further additions or deletions. Third, we verified the OWL$_{\text{strict}}$ versions by validating them against an external SHACL validator service[25]. Fourth, we imported the OWL$_{\text{strict}}$ versions into the PoolParty ontology editor and verified the data in the user interface. Table 2 shows the details for the ontologies and their OWL$_{\text{strict}}$ versions. We can see an increase in size for both OWL$_{\text{strict}}$ versions of the ontologies with repairs establishing clear and well-defined descriptions. SemOpenAlex already satisifed the constraints, so no changes were done. The ontology RDF files are published at[26].

Ontology	Initial size	Additions	Deletions	OWL$_{\text{strict}}$ size	Conforms
DITA	490	104	9	585	yes
gist core	3060	50	9	3101	yes
ERA	7163	69	54	7178	yes
SemOpenAlex	848	0	0	848	yes
EBUCorePlus	11930	2044	27	13947	yes
DPROD	147	11	7	151	yes

Table 2. Statistics for OWL$_{\text{strict}}$ ontologies.

6 Conclusions & Future Work

In this paper we presented OWL$_{\text{strict}}$, an OWL fragment designed to avoid ambiguities in OWL ontologies by requiring well-defined ontology entities and syntactic restrictions to be satisfied. Next, we expressed the restrictions using SHACL to check ontology RDF graphs for conformance with OWL$_{\text{strict}}$. For practical purposes, motivated by reuse of existing ontologies on the Web, we applied our recent work on SHACL repairs, which enables us to modify arbitrary ontologies to conform to OWL$_{\text{strict}}$. We performed an evaluation with six ontologies we use in real-world projects, created OWL$_{\text{strict}}$ versions using SHACL repairs, and verified conformance with a SHACL processor. Future work will explore different preferences in repairing ontologies for OWL strict. For example, one might want to avoid removing *rdfs:subClassOf* relations or prefer adding ObjectProperty for Property nodes, as described in [25]. Generally, we will consider OWL semantics for the repairs as well. With our work, we provided an approach to avoid ambiguities in ontology RDF graphs, thereby making it easier for tools to work with OWL ontologies in practice.

[24] `https://ekgf.github.io/dprod/`
[25] `https://www.itb.ec.europa.eu/shacl/any/upload`
[26] `https://github.com/robert-david/OWL-strict/tree/main/examples`

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) [10.55776/COE12] (Polleres); the Austrian Science Fund (FWF) and netidee SCIENCE [T1349-N], and the Vienna the Science and Technology Fund (WWTF) [10.47379/ICT2201] (Ahmetaj).

References

1. Ahmetaj, S., David, R., Polleres, A., Šimkus, M.: Repairing SHACL Constraint Violations Using Answer Set Programming. In: The Semantic Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings. p. 375–391. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-19433-7_22
2. Ahmeti, A., Fernández, J.D., Polleres, A., Savenkov, V.: Updating Wikipedia via DBpedia mappings and SPARQL. In: The Semantic Web: 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28–June 1, 2017, Proceedings, Part I 14. pp. 485–501. Springer (2017)
3. Allemang, D., Hendler, J.: Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edn. (2011)
4. Baader, F., Koopmann, P., Kriegel, F., Nuradiansyah, A.: Optimal ABox Repair w.r.t. Static EL TBoxes: From Quantified ABoxes Back To ABoxes. In: The Semantic Web: 19th International Conference, ESWC 2022, Hersonissos, Crete, Greece, May 29 – June 2, 2022, Proceedings. p. 130–146. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-06981-9_8
5. Ben-David, S., Trefler, R., Weddell, G.: Model checking using description logic. Journal of Logic and Computation **20**(1), 111–131 (2010)
6. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press, Cambridge, MA, USA (2000)
7. Corman, J., Florenzano, F., Reutter, J.L., Savković, O.: Validating SHACL constraints over a SPARQL endpoint. In: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I 18. pp. 145–163. Springer (2019)
8. Day, D., Priestley, M., Schell, D.: Introduction to the Darwin Information Typing Architecture. IBM corporation (2005)
9. De Bruijn, J., Lara, R., Polleres, A., Fensel, D.: OWL DL vs. OWL Flight: Conceptual modeling and reasoning for the Semantic Web. In: Proceedings of the 14th international conference on World Wide Web. pp. 623–632 (2005)
10. Galkin, M., Auer, S., Vidal, M.E., Scerri, S.: Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems. In: ICEIS (2). pp. 88–98 (2017)
11. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: Yet to arrive on the Web of Data? arXiv preprint arXiv:1202.0984 (2012)
12. Haase, P., Herzig, D.M., Kozlov, A., Nikolov, A., Trame, J.: metaphactory: A platform for knowledge graph management. Semantic Web **10**(6), 1109–1125 (2019). https://doi.org/10.3233/SW-190360, https://doi.org/10.3233/SW-190360

13. Haller, A., Fernández, J.D., Kamdar, M.R., Polleres, A.: What are Links in Linked Open Data? A Characterization and Evaluation of Links between Knowledge Graphs on the Web. ACM Journal of Data and Information Quality (JDIQ) **2**(2), 1—-34 (May 2020). https://doi.org/10.1145/3369875, pre-print available at https://epub.wu.ac.at/7193/
14. Hogan, A.: Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora. Ph.D. thesis, Digital Enterprise Research Institute (2011)
15. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the Pedantic Web. In: 3rd International Workshop on Linked Data on the Web (LDOW2010) at WWW2010. Raleigh, USA (Apr 2010), http://aidanhogan.com/docs/pedantic_ldow10.pdf
16. Horridge, M., Bechhofer, S.: The OWL API: A Java API for Working with OWL 2 Ontologies. In: OWLED. vol. 529, pp. 11–21 (2009)
17. ter Horst, H.J.: Combining RDF and Part of OWL with Rules: Semantics, Decidability, Complexity. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3729, pp. 668–684. Springer (2005). https://doi.org/10.1007/11574620_48, https://doi.org/10.1007/11574620_48
18. Kalyanpur, A.: Debugging and repair of owl ontologies. Ph.D. thesis, University of Maryland at College Park, USA (2006), AAI3222483
19. Knublauch, H., Kontokostas, D.: Shapes constraint language (SHACL). Tech. rep., W3C (Jul 2017), https://www.w3.org/TR/shacl/
20. Li, Y., Lambrix, P.: Repairing EL ontologies using weakening and completing. In: European Semantic Web Conference. pp. 298–315. Springer (2023)
21. Michael Uschold, D.M.: Introduction to gist. Tech. rep., Semantic Arts, Inc. (01 2013), https://iaoa.org/isc2014/uploads/Whitepaper-Uschold-IntroductionToGist.pdf
22. Motik, B., Horrocks, I., Sattler, U.: Adding Integrity Constraints to OWL. In: OWLED. vol. 258 (2007)
23. Munoz, S., Pérez, J., Gutierrez, C.: Simple and efficient minimal RDFS. Journal of web semantics **7**(3), 220–234 (2009)
24. Rojas, J.A., Aguado, M., Vasilopoulou, P., Velitchkov, I., Van Assche, D., Colpaert, P., Verborgh, R.: Leveraging semantic technologies for digital interoperability in the european railway domain. In: International Semantic Web Conference. pp. 648–664. Springer (2021)
25. Schneider, M.: OWL 2 Web Ontology Language: RDF-Based Semantics. Tech. rep., W3C (10 2009), https://www.w3.org/TR/owl2-rdf-based-semantics/
26. Sirin, E.: Data validation with OWL integrity constraints. In: Web Reasoning and Rule Systems: Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings 4. pp. 18–22. Springer (2010)