

# Characterizing and Detecting Integrity Issues in OWL Instance Data

Jiao Tao, Li Ding, Jie Bao, and Deborah L. McGuinness

Tetherless World Constellation, Computer Science Department  
Rensselaer Polytechnic Institute  
110 8th St., Troy, NY 12180  
{taoj2,ding1,baojie,dlm}@cs.rpi.edu

**Abstract.** OWL instance data evaluation, identified by our previous work, is to check whether the instance data conforms to the semantic expectation of its targeting applications. We have identified some integrity issues raised by applications demanding closed world reasoning. In this paper, we present a formal characterization of those integrity issues using autoepistemic operators. This approach allows users to selectively enforce integrity constraints on individual classes and properties, thus allows not only global but also local closed world reasoning with OWL instance data. We also show a practical SPARQL-based approach that is a sound approximation for detecting integrity issues.

**Key words:** Instance Data, Evaluation, Autoepistemic Description Logics, OWL, SPARQL

## 1 Introduction

Semantic Web data can be classified into terminological ontologies, i.e., schema statements, and instance data [8]. Before using Semantic Web data, it is important for users to verify whether certain instance data meets the semantic expectations of the ontology designer with particular application requirements in mind [1]. Our previous work has recognized this task as Semantic Web instance data evaluation and identified some integrity issues beyond conventional syntactic errors and logical inconsistencies [1]. The presence of integrity issues depends on assumptions hold by applications as illustrated in the following example.

**Example 1** :The wine ontology<sup>1</sup> defines an `owl:cardinality` restriction (=1) on the property `wine:hasColor` over the class `wine:Wine`, i.e., each wine instance is expected to have exactly one color. This restriction may be either ignored or enforced in different applications: (i) when a wine ontology is used by a wine store owner that needs to arrange his wine inventory by color, the presence of the color statement of a wine is always required and a missing color value should be warned as an error in the data; and (ii) when a tourist describes in a blog article that he drank wine at a nice restaurant, readers of the article may not necessarily be interested to know the color of the wine, which, although is missing in the article, may be given somewhere else.

<sup>1</sup> Wine ontology: <http://www.w3.org/TR/owl-guide/wine>

The difference in the expected semantics of ontology data as described above can be captured by adopting the closed world assumptions (CWA, as in (i) ) or open world assumptions (OWA, as in (ii)). The key difference between the two assumptions is that under CWA, any fact that is not specified is assumed to be false, such that we only know the knowledge given in the knowledge base; on the other hand, under OWA, a missing fact is not necessarily false and may be given or inferred in other ways. Our previous work attempts to capture integrity issues in OWL instance data by adopting global closed world assumptions. Based on these assumptions, all semantic restrictions encoded in referenced ontologies are checked as integrity constraints while evaluating instance data.

In this work, we further extend the framework to also support local closed world assumptions when they are required by applications. In such applications, a user may have complete knowledge about part of the domain (e.g., the wine color for a wine store owner) while has no complete knowledge or is not interested in other things in the domain (e.g., the wine flavor for the wine store owner, as it is not needed for inventory maintenance). To address above issues, we use Autoepistemic Description Logics (ADLs) [4] to characterize the typical integrity issues as integrity constraints. This paper presents some initial steps in formalizing integrity issues in OWL instance data as well as a theoretically sound approach to pinpoint the causes of such issues.

Note that there is also an alternative approach to represent integrity constraints in OWL by dividing TBox axioms into standard axioms and constraint axioms [9]. We adopt the ADLs approach as there exists a straightforward, scalable, and sound implementation for the ADLs approach, while the approach in [9] requires some rather complicated operations, e.g., skolemization, that may not be practical at the web-scale. A sound implementation is highly desirable as completeness is typically not critical, or not even feasible in practical time, in the context of OWL instance data evaluation on large data sets.

The main contributions of the paper include:

- We use Autoepistemic Description Logics (ADLs) to characterize the integrity issues as integrity constraints. This formal representation has two major advantages: (i) clarifying the semantics of integrity issues which were previously captured by RDF graph patterns in our previous work, and (ii) enabling local closed world reasoning which is more flexible than the previous global closed world reasoning.
- While reasoning with autoepistemic extensions to OWL is still a hard problem, we show that issues that are characterized as ADLs integrity constraints can be soundly approximated by SPARQL graph patterns. Furthermore, this SPARQL-based approach allows us not only to detect but also to pinpoint the causes of those issues.

The rest of the paper is organized as follows: Section 2 briefly introduces the semantics of ADLs and SPARQL; Section 3 characterizes integrity issues in OWL instance data as integrity constraints using ADLs; Section 4 shows how to detect the integrity issues by SPARQL queries in a sound manner; Section 5 discusses

the limitations and possible extensions of the basic ADLs in representing some integrity issues; Section 6 concludes this work and points out future direction.

## 2 Preliminary

### 2.1 Autoepistemic Description Logics

Autoepistemic Description Logics (ADLs)[3, 4, 2] extends DLs with two modal operators **K** and **A** from the nonmonotonic logic MKNF [6, 7]. Intuitively, **K** and **A** operators refer to minimal knowledge and assumptions respectively. The basic ADLs language is  $\mathcal{ALCC}_{NF}$  that extends the DL  $\mathcal{ALC}$  by incorporating the **K** and **A**. The syntax of  $\mathcal{ALCC}_{NF}$  is as follows:

$$\begin{aligned} C &\longrightarrow \top | \perp | C_a | C \sqcap C | C \sqcup C | \neg C | \exists R.C | \forall R.C | \mathbf{K}C | \mathbf{A}C \\ R &\longrightarrow R_a | \mathbf{K}R_a | \mathbf{A}R_a \end{aligned}$$

Where  $C_a$  denotes a primitive concept,  $C$  denotes a concept,  $R_a$  denotes a primitive role, and  $R$  denotes a role.

An epistemic interpretation is defined as a triple  $(\mathcal{I}, \mathcal{X}, \mathcal{Y})$  where  $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  is a first-order interpretation defined over domain  $\Delta^{\mathcal{I}}$  and an interpretation function  $\cdot^{\mathcal{I}}$ , and  $\mathcal{X}, \mathcal{Y}$  are sets of first-order interpretations (i.e., possible worlds). The semantics of  $\mathcal{ALCC}_{NF}$  is given as follows:

$$\begin{aligned} \top^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \Delta^{\mathcal{I}} \\ \perp^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \emptyset \\ (C_1 \sqcap C_2)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= C_1^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \cap C_2^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \\ (C_1 \sqcup C_2)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= C_1^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \cup C_2^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \\ (\neg C)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \\ (\exists R.C)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \{s \in \Delta^{\mathcal{I}} | \exists t. (s, t) \in R^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \wedge t \in C^{\mathcal{I}, \mathcal{X}, \mathcal{Y}}\} \\ (\forall R.C)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \{s \in \Delta^{\mathcal{I}} | \forall t. (s, t) \in R^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \rightarrow t \in C^{\mathcal{I}, \mathcal{X}, \mathcal{Y}}\} \\ (\mathbf{K}C)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \bigcap_{\mathcal{J} \in \mathcal{X}} C^{\mathcal{J}, \mathcal{X}, \mathcal{Y}} \\ (\mathbf{A}C)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \bigcap_{\mathcal{J} \in \mathcal{Y}} C^{\mathcal{J}, \mathcal{X}, \mathcal{Y}} \\ (\mathbf{K}R)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \bigcap_{\mathcal{J} \in \mathcal{X}} R^{\mathcal{J}, \mathcal{X}, \mathcal{Y}} \\ (\mathbf{A}R)^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} &= \bigcap_{\mathcal{J} \in \mathcal{Y}} R^{\mathcal{J}, \mathcal{X}, \mathcal{Y}} \end{aligned}$$

$C \sqsubseteq D$  is satisfied by an epistemic interpretation  $(\mathcal{I}, \mathcal{X}, \mathcal{Y})$  if  $C^{\mathcal{I}, \mathcal{X}, \mathcal{Y}} \subseteq D^{\mathcal{I}, \mathcal{X}, \mathcal{Y}}$ .  $C(a)$ (or  $R(a, b)$ ) is satisfied by an epistemic interpretation  $(\mathcal{I}, \mathcal{X}, \mathcal{Y})$  if  $a^{\mathcal{I}} \in C^{\mathcal{I}, \mathcal{X}, \mathcal{Y}}$  (or  $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}, \mathcal{X}, \mathcal{Y}}$ ). An epistemic model  $\mathcal{M}$  for an  $\mathcal{ALCC}_{NF}$  knowledge base  $\Sigma$  is a non-empty set of first-order interpretations, and for each  $\mathcal{I} \in \mathcal{M}$ ,  $(\mathcal{I}, \mathcal{M}, \mathcal{M})$  satisfies all of the axioms in  $\Sigma$ , and for any set  $\mathcal{N}$  of first order interpretations such that  $\mathcal{M} \subset \mathcal{N}$ ,  $(\mathcal{I}, \mathcal{N}, \mathcal{M})$  does not satisfy  $\Sigma$ .  $\Sigma$  is satisfiable if there exists an epistemic model for it.  $\Sigma$  implies an axiom  $\alpha$  ( $\Sigma \models \alpha$ ) if  $\alpha$  is satisfied in every model of  $\Sigma$ .

Note that the difference between **K** and **A** is that we have different requirements for the possible worlds in  $\mathcal{X}$  and  $\mathcal{Y}$ . We are only interested in the epistemic models that the  $\mathcal{X}$  is *maximized* (minimal knowledge) and the  $\mathcal{Y}$  contains only assumptions which are justified by the facts in knowledge base.

[5] have proposed the extension of OWL, corresponding to the DL  $\mathcal{SHOIN}(\mathcal{D})$ , by incorporating autoepistemic operators and have shown that with the additional introspective nature, OWL could be used to capture some non-monotonic features which are needed by many Semantic Web applications.

## 2.2 Syntax and Semantics of SPARQL

SPARQL is a W3C recommended RDF query language. In this section we summarize its syntax and semantics as provided in [10].

An RDF triple is a subject-predicate-object tuple  $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$  where  $I$  denotes IRIs,  $B$  denotes blank nodes,  $L$  denotes literals. Let  $IL$  denotes  $I \cup L$ ,  $T$  denotes  $I \cup B \cup L$ ,  $V$  denotes an infinite set of variables disjoint from the sets mentioned before. An RDF graph (or RDF dataset) is a set of RDF triples. A SPARQL built-in condition is an expression composed by operators  $\neg, \wedge, \vee, <, \leq, >, \geq, =$ , and unary predicates `bound`, `isBlank`, `isIRI`, over  $V \cup IL$  and constants.

A SPARQL graph pattern expression is defined as follows:

- (1) A tuple from  $(IL \cup V) \times (I \cup V) \times (IL \cup V)$
- (2)  $(P_1 \text{ AND } P_2), (P_1 \text{ OPT } P_2), (P_1 \text{ UNION } P_2)$  are graph patterns if  $P_1$  and  $P_2$  are graph patterns.
- (3)  $(P \text{ FILTER } R)$  is a graph pattern if  $P$  is a graph pattern and  $R$  is a SPARQL built-in condition.

A mapping  $\mu$  is a partial function  $V \rightarrow T$ . The mapping  $\mu$  on triple pattern  $t$ ,  $\mu(t)$ , is computed by replacing the variables in  $t$  according to  $\mu$ . We say that two mappings are compatible if  $\mu_1(x) = \mu_2(x)$  when  $x \in \text{domain}(\mu_1) \cap \text{domain}(\mu_2)$ .

The join, union, difference, and left outer-join operation on two set of mappings  $\Omega_1$  and  $\Omega_2$  are defined as follows:

- (1)  $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ are compatible}\}$
- (2)  $\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\}$
- (3)  $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \text{for all } \mu_2 \in \Omega_2, \mu_1 \text{ and } \mu_2 \text{ are not compatible}\}$
- (4)  $\Omega_1 \bowtie \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$

A Mapping  $\mu$  satisfies built-in condition  $R$ ,  $\mu \models R$ , if  $R$  is `bound(?X)` and  $?X \in \text{domain}(\mu)$ . The operators  $=, \neg, \vee, \wedge$  in  $R$  are interpreted in the usual way as in first-order logic.

The evaluation of a graph pattern over a dataset  $D$ ,  $[[\cdot]]_D$ , is defined as follows:

- (1)  $[[t]]_D = \{\mu \mid \text{domain}(\mu) = \text{var}(t) \text{ and } \mu(t) \in D\}$ , where  $\text{var}(t)$  denotes all of the variables in  $t$ .
- (2)  $[[ (P_1 \text{ AND } P_2) ] ]_D = [[P_1]]_D \bowtie [[P_2]]_D$
- (3)  $[[ (P_1 \text{ OPT } P_2) ] ]_D = [[P_1]]_D \bowtie \cup [[P_2]]_D$
- (4)  $[[ (P_1 \text{ UNION } P_2) ] ]_D = [[P_1]]_D \cup [[P_2]]_D$
- (5)  $[[ (P \text{ FILTER } R) ] ]_D = \{\mu \in [[P]]_D \mid \mu \models R\}$

where  $t$  is a triple pattern,  $P_1$  and  $P_2$  are graph patterns,  $\mu$  is a mapping,  $R$  is a built in condition, and  $D$  is an RDF dataset over  $T$ .

## 3 Characterizing Integrity Issues in OWL Instance Data

In this section, we show how to model some typical integrity issues in OWL instance data using ADLs. Here we only discuss issues that can be captured by  $\mathcal{ALCK}_{\mathcal{NF}}$ , some other issues will be discusses in Section 5.

In what follows, by default, we use  $KB, \mathcal{T}, \mathcal{A}, IC, C(D), P$  to denote a knowledge base, TBox, ABox, integrity constraint, class, and property respectively.

### 3.1 Missing Property Value Issues (MPV)

Missing property value issues may arise when a property value that is expected to be specified is not explicitly given in the data set. We identify three MPV issues  $MPV_{\exists}$ ,  $IC_{MPV_{=}}$ , and  $IC_{MPV_{\geq}}$  corresponding to the OWL constructs `owl:someValuesFrom`, `owl:cardinality` and `owl:minCardinality`. The first case is represented here. The latter two cases are discussed in Section 5.

**Definition 1** (*MPV<sub>∃</sub> Issue*) *Given a knowledge base  $KB = \{\mathcal{T}, \mathcal{A}\}$  which is satisfiable. Let  $IC_{MPV_{\exists}} = \{KC \sqsubseteq \exists AP.\top\}$  for some  $C, P$ . If  $\{\mathcal{T}, \mathcal{A}, IC_{MPV_{\exists}}\}$  is not satisfiable, then the ABox  $\mathcal{A}$  has a MPV<sub>∃</sub> issue.*

**Example 2** :Assume that there exist `Individual(W type(Wine))` and `Class(Wine partial restriction(locatedIn someValuesFrom(Region)))` in the instance data and the wine ontology respectively. The application requires that each wine instance to have a location, thus integrity constraint  $\{KWine \sqsubseteq \exists AlocatedIn.\top\}$  is added. If the constraint is not satisfied, a MPV-∃ issue would occur.

### 3.2 Unexpected Individual Type Issues (UIT)

Unexpected individual type issues may occur when a given individual in the instance data is declared to have types that are not expected by the referenced ontologies, or is missing a type declaration when it is expected. We list three UIT issues  $UIT_d$ ,  $UIT_r$ , and  $UIT_{\forall}$  corresponding to the RDFS/OWL constructs `rdfs:domain`, `rdfs:range` and `owl:allValuesFrom`.

**Definition 2** (*UIT<sub>d</sub> Issue*) *Given a knowledge base  $KB = \{\mathcal{T}, \mathcal{A}\}$  which is satisfiable. Let  $IC_{UIT_d} = \{\exists KP.\top \sqsubseteq AC\}$  for some  $C, P$ . If  $\{\mathcal{T}, \mathcal{A}, IC_{UIT_d}\}$  is not satisfiable, then the ABox  $\mathcal{A}$  has a UIT<sub>d</sub> issue.*

**Example 3** :Assume that there exist `Individual(W value(hasColor Red))` and `ObjectProperty(hasColor domain(Wine))` in the wine instance data and wine ontology respectively, but no other facts about W is given. Each individual having a value for property `hasColor` is required to have a type `Wine`, i.e.,  $\{\exists KhasColor.\top \sqsubseteq AWine\}$ , otherwise a UIT<sub>d</sub> issue would arise in this example.

Similarly, we can define UIT issues for `rdfs:range` and `owl:allValuesFrom` as follows:

**Definition 3** (*UIT<sub>r</sub> Issue*) *Given a knowledge base  $KB = \{\mathcal{T}, \mathcal{A}\}$  which is satisfiable. Let  $IC_{UIT_r} = \{\top \sqsubseteq \forall P.AC\}$  for some  $C, P$ . If  $\{\mathcal{T}, \mathcal{A}, IC_{UIT_r}\}$  is not satisfiable, then the ABox  $\mathcal{A}$  has a UIT<sub>r</sub> issue.*

**Definition 4** (*UIT<sub>∃</sub> Issue*) *Given a knowledge base  $KB = \{\mathcal{T}, \mathcal{A}\}$  which is satisfiable. Let  $IC_{UIT_{\forall}} = \{KC \sqsubseteq \forall P.AD\}$  for some  $C, D, P$ . If  $\{\mathcal{T}, \mathcal{A}, IC_{UIT_{\forall}}\}$  is not satisfiable, then the ABox  $\mathcal{A}$  has a UIT<sub>∃</sub> issue.*

### 3.3 Non-specific Individual Type Issues (NSIT)

Non-specific individual type issues may arise if a given individual in the instance data is declared to have a general type rather than a more specific type that is expected by the application.

**Definition 5 (NSIT Issue)** *Given a knowledge base  $KB = \{T, \mathcal{A}\}$  which is satisfiable. Let  $IC_{NSIT} = \{KC \sqsubseteq AC_1 \sqcup \dots \sqcup AC_n\}$ , for some  $\{C, C_1, \dots, C_n\}$ . If  $\{T, \mathcal{A}, IC_{NSIT}\}$  is not satisfiable, then the ABox  $A$  has a NSIT issue.*

**Example 4 :**In the wine ontology, class `WineTaste` has three subclasses: `Body`, `Flavor`, and `Sugar`. An applications expects that if an individual  $T$  is declared to have the type `WineTaste`, then  $T$  is expected to have more specific types which are from  $\{\text{Body}, \text{Flavor}, \text{Sugar}\}$ , i.e.,  $\{\mathbf{KWineTaste} \sqsubseteq \mathbf{ABody} \sqcup \mathbf{AFlavor} \sqcup \mathbf{ASugar}\}$ , otherwise a NSIT issue would occur in the instance data.

To summarize, most typical integrity issues in OWL instance data can be characterized as integrity constraints using ADLs. Please note that:

(i) Usually each integrity constraint corresponds to a similar inclusion axiom in the TBox, for instance,  $\{\exists \mathbf{KhasColor}. \top \sqsubseteq \mathbf{AWine}\}$  corresponds to  $\exists \text{hasColor}. \top \sqsubseteq \text{Wine}$ . However, this is not necessarily the case for all integrity constraints.

(ii) On the other hand, an application may not require that all TBox axioms to have corresponding integrity constraints. Constraints can be selectively added to the knowledge base, thus allow local rather than global closed world reasoning.

## 4 SPARQL-based Approximation to Integrity Issues Detection

In general, reasoning in ADLs is a hard problem. Francesco M. Donini and others [4] have presented that in  $\mathcal{ALCK}_{\mathcal{NF}}$  a simple KB, in which there is no occurrence of epistemic operators in the scope of quantifiers, can be reduced to a subjectively quantified ABox, in which the instance checking problem is decidable. However, in more expressive ADLs, the decidability and complexity results are still unknown. Stephan Grimm and others [5] have proposed extending OWL with autoepistemic operators  $\mathcal{K}$  and  $\mathcal{A}$  and formalizing integrity constraints using expressive ADLs. However, no corresponding reasoning algorithms provided for the proposal in [5]. To the best of our knowledge, there is still no reasoner that can handle the reasoning in the autoepistemic extension of OWL.

To detect integrity issues in OWL instance data characterized by ADLs, we adopt a practical SPARQL-based approach that can generate results in a sound but not necessarily complete manner, that is, what the approach discovered are the issues (but may not be all of them) captured by the ADLs integrity constraints. Furthermore, the approach is easy to implement using existing tools.

In this section, we prove that our SPARQL-based issue detection mechanism is a sound solution to the integrity issue detection problem. We use  $MPV_{\exists}$  issue as an example. Other types of issues can be handled in similar manner.

**Definition 6 (Integrity Violation)** Given a knowledge base  $KB = \{\mathcal{T}, \mathcal{A}\}$ , where  $\mathcal{T}$  and  $\mathcal{A}$  denotes TBox and ABox respectively. Let  $\mathcal{A}|_i = \{\alpha \mid \alpha \in \mathcal{A} \text{ and } i \text{ occurs in } \alpha\}$  where  $i$  is an individual in  $\mathcal{A}$ . If  $\{\mathcal{T}, \mathcal{A}|_i, IC\}$  is not satisfiable, then  $i$  is said to violate  $IC$ , denoted by  $i \not\models IC$ .

**Definition 7 ( $MPV_{\exists}$  SPARQL Pattern)** Let the SPARQL graph pattern  $P_{MPV_{\exists}}$  be:

```
?i rdf:type C
OPT (?i P ?o)
FILTER (!BOUND(?o))
```

Now we prove that using the  $MPV_{\exists}$  SPARQL pattern, we can detect individuals that are involved in  $MPV_{\exists}$  issues as defined by ADL.

**Theorem 1** If an individual  $i$  is contained in the evaluation of the graph pattern  $P_{MPV_{\exists}}$  over  $\mathcal{A}$ , i.e.,  $i \in [[P_{MPV_{\exists}}]]_{\mathcal{A}}$ , then  $i \not\models IC_{MPV_{\exists}}$ .

**Proof:** The proof has two steps: first we prove that the SPARQL pattern  $P_{MPV_{\exists}}$  can find a set of individuals in the ABox that are missing (expected) property values, then we prove that those individuals violate the integrity constraint  $IC_{MPV_{\exists}} = \{KC \sqsubseteq \exists AP.T\}$ .

(1) We now prove that if an individual  $i$  appears in the evaluation of the graph pattern  $P_{MPV_{\exists}}$  over  $\mathcal{A}$ , then  $C(i) \in \mathcal{A}$  and  $\nexists j.P(i, j) \in \mathcal{A}$ .

Proof: Let  $P_1 = (?i \text{ rdf:type } C)$ ,  $P_2 = (?i \text{ P } ?o)$ ,  $R = (!BOUND(?o))$ , Then

$$\begin{aligned}
[[P_1]]_{\mathcal{A}} &= \{x \mid C(x) \in \mathcal{A}\} \\
[[P_2]]_{\mathcal{A}} &= \{(x, y) \mid P(x, y) \in \mathcal{A}\} \\
[[P_1]]_{\mathcal{A}} \bowtie [[P_2]]_{\mathcal{A}} &= \{(x, y) \mid C(x) \in \mathcal{A} \text{ and } P(x, y) \in \mathcal{A}\} \\
[[P_1]]_{\mathcal{A}} \setminus [[P_2]]_{\mathcal{A}} &= \{x \mid C(x) \in \mathcal{A} \text{ and } \nexists y.P(x, y) \in \mathcal{A}\} \\
[[P_1 \text{ OPT } P_2]]_{\mathcal{A}} &= [[P_1]]_{\mathcal{A}} \bowtie [[P_2]]_{\mathcal{A}} \\
&= ([[P_1]]_{\mathcal{A}} \bowtie [[P_2]]_{\mathcal{A}}) \cup ([[P_1]]_{\mathcal{A}} \setminus [[P_2]]_{\mathcal{A}}) \\
&= \{(x, y) \mid C(x) \in \mathcal{A} \text{ and } P(x, y) \in \mathcal{A}\} \cup \\
&\quad \{x \mid C(x) \in \mathcal{A} \text{ and } \nexists y.P(x, y) \in \mathcal{A}\}
\end{aligned}$$

Since,

$$\begin{aligned}
\mu \in \{(x, y) \mid C(x) \in \mathcal{A} \text{ and } P(x, y) \in \mathcal{A}\} &\rightarrow (\mu \not\models R) \\
\mu \in \{x \mid C(x) \in \mathcal{A} \text{ and } \nexists y.P(x, y) \in \mathcal{A}\} &\rightarrow (\mu \models R)
\end{aligned}$$

We got,

$$\begin{aligned}
[[P]]_{\mathcal{A}} &= [[(P_1 \text{ OPT } P_2) \text{ FILTER } R]]_{\mathcal{A}} \\
&= \{\mu \in [[(P_1 \text{ OPT } P_2)]]_{\mathcal{A}} \mid \mu \models R\} \\
&= \{x \mid C(x) \in \mathcal{A} \text{ and } \nexists y.P(x, y) \in \mathcal{A}\}
\end{aligned}$$

Thus, if  $i$  appears in  $[[P_{MPV_{\exists}}]]_{\mathcal{A}}$ , we can conclude that  $C(i) \in \mathcal{A}$  and  $\nexists j.P(i, j) \in \mathcal{A}$ .

(2) Let  $i$  be an instance, if  $C(i) \in \mathcal{A}$  and  $\nexists j.P(i, j) \in \mathcal{A}$ , then  $i \not\models IC_{MPV_{\exists}}$ , thus  $\mathcal{A}$  has a  $MPV_{\exists}$  issue.

Proof: If  $C(i) \in \mathcal{A}$  and  $\nexists j.P(i, j) \in \mathcal{A}$ , we know that  $C(i) \in \mathcal{A}|_i$  and  $\nexists j$  such that  $P(i, j) \in \mathcal{A}|_i$ . The knowledge base  $K' = \{\mathcal{T}, \mathcal{A}|_i, IC_{MPV_{\exists}}\}$  is not

constant because there is no epistemic model for it. We prove that by contradiction. Suppose  $K'$  has an epistemic model  $\mathcal{M}$ , then for all  $\mathcal{I} \in \mathcal{M}$ , we have  $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \mathbf{KC}(i)$ , then by  $IC_{MPV_{\exists}}$ ,  $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \exists \mathbf{AP}.\top(i)$ , therefore  $\exists j \in \Delta^{\mathcal{I}}$  such that  $(i, j) \in P^{\mathcal{J}, \mathcal{M}, \mathcal{M}}$  for all  $\mathcal{J} \in \mathcal{M}$ . However, for every  $\mathcal{M}' \supset \mathcal{M}$ ,  $(\mathcal{I}, \mathcal{M}', \mathcal{M})$  also satisfies  $K'$ , thus  $M$  is not maximal, therefore  $M$  is not an epistemic model of  $K'$ .  $\square$

Theorem 1 means that the SPARQL graph pattern  $P_{MPV_{\exists}}$  provides a sound solution to the MPV issue which is formalized as integrity constraint  $IC_{MPV_{\exists}}$  using ADLs, since all individuals discovered by the pattern are causes of MPV issues. Similarly, we have also provided SPARQL solutions to other typical integrity issues in [12]. A strong advantage of the SPARQL-based solutions is that it is easy to implement using exiting tools. We have implemented the proposed algorithm using Java, Jena<sup>2</sup>, and Pellet<sup>3</sup>. Our OWL instance data evaluation service<sup>4</sup> can detect typical integrity issues in instance data, in addition to syntax errors and logical inconsistencies.

## 5 Discussion

In section 3, we have presented how to model some integrity issues as integrity constraints with  $\mathcal{ALCK}_{NF}$ . Although these integrity constraints can capture some representative issues encountered in using OWL instance data,  $\mathcal{ALCK}_{NF}$  is not expressive enough to characterize the other typical issues we have listed in [1]. This is because  $\mathcal{ALCK}_{NF}$  only extends  $\mathcal{ALC}$  with autoepistemic operators, while OWL corresponds to the DL  $\mathcal{SHOIN}(\mathcal{D})$  which is strictly more expressive than  $\mathcal{ALC}$ . In this section, we discuss the need for extensions of  $\mathcal{ALCK}_{NF}$  such that the other integrity issues may also be modeled with ADLs.

Please note that some important characteristics of those extensions to  $\mathcal{ALCK}_{NF}$ , e.g., decidability or complexity, are still unknown. However, it is promising to design a sound algorithm based on the SPARQL approximation approach we applied in this paper. We will study such an algorithm in future work.

### 5.1 Missing Property Value Issues (MPV)

Besides the  $MPV_{\exists}$  case presented in Section 3, there are other two cases of MPV issues corresponding to OWL constructs `owl:cardinality` and `owl:minCardinality`.

**MPV<sub>=</sub> Issue:** `owl:cardinality` restricts the number of values on certain properties an instance can have. To model this restriction, the quantification constructor  $\mathcal{Q}$  is needed in ADLs. With this extension, given `Class(C partial restriction(P cardinality(n)))`, the corresponding integrity constraint is  $IC_{MPV_{=}} = \{\mathbf{KC} \sqsubseteq (= n)\mathbf{AP}.\top\}$ . Given a knowledge base  $\text{KB} = \{\mathcal{T}, \mathcal{A}\}$  which is satisfiable, if  $\{\mathcal{T}, \mathcal{A}, IC_{MPV_{=}}\}$  is not satisfiable, the the ABox  $\mathcal{A}$  has a  $MPV_{=}$  issue.

**MPV<sub>≥</sub> Issue:** Similarly, `owl:minCardinality` corresponds to the integrity constraint  $IC_{MPV_{\geq}} = \{\mathbf{KC} \sqsubseteq (\geq n)\mathbf{AP}.\top\}$ . A  $MPV_{\geq}$  issue is defined correspondingly.

<sup>2</sup> Jena Semantic Web Framework, <http://jena.sourceforge.net/>

<sup>3</sup> Pellet: The Open Source OWL DL Reasoner, <http://pellet.owldl.com/>

<sup>4</sup> <http://onto.rpi.edu/demo/oie/>

## 5.2 Excessive Property Value Issues (EPV)

Excessive property value issues may arise if more property values than expected are explicitly declared in the data set. In particular, there are two cases corresponding to OWL constructs `owl:cardinality` and `owl:maxCardinality`. With the  $\mathcal{Q}$  extension, we can model the integrity constraints as follows:

$$\begin{aligned} IC_{EPV=} &= \{\mathbf{KC} \sqsubseteq (= n)\mathbf{AP}.\top\} \\ IC_{EPV\leq} &= \{\mathbf{KC} \sqsubseteq (\leq n)\mathbf{AP}.\top\} \end{aligned}$$

## 5.3 Redundant Individual Type Issue (RIT)

Redundant individual type issues may happen if an individual is explicitly declared to have both  $C$  and a  $C$ 's superclass as its types. Assume that an individual  $W$  is explicitly declared to have types `Zinfandel` and `Wine`. Actually, the "Wine" type declaration is redundant since it can be inferred from the fact that `Zinfandel`  $\sqsubseteq$  `Wine`. To address this issue, we have to distinguish the explicitly told facts from those facts that are inferred from the facts in the knowledge base. A potential solution to this problem is to use two epistemic operators to refer to the two cases respectively, and bring them into ADLs.

## 5.4 Uniqueness Issues (UT)

The uniqueness issues may arise if an instance that is expected to be unique has two or more individuals in the data set. In particular, there are following two cases of UT issues:

**Nominal:** A nominal (the DL feature " $\mathcal{O}$ ") is expected to have only one instance. Thus, given a nominal  $o$ , we may add integrity constraints:

$IC_{UT_o} = \{\mathbf{K}o \sqsubseteq \exists R^-.n; \mathbf{K}n \sqsubseteq (= 1R.\mathbf{A}o)\}$ , where  $n$  is a new nominal with single instance and  $R$  a new property. Both  $n$  and  $R$  do not exist in the original KB.

**Key:** A key requirement can be specified by `owl:inverseFunctionalProperty` (with DL features " $\mathcal{T}$ " and " $\mathcal{F}$ "), e.g., `inverseFunctionalProperty(hasEmail)` will enforce that two persons that have the same email to be actually the same person. To model this as an integrity constraint, given a property  $R$ , we may have the following integrity constraints:

$$IC_{UT_k} = \{\top \sqsubseteq (\leq 1)\mathbf{A}P^-. \top\} \text{ for some } P.$$

## 6 Conclusions and Future Work

This work formally investigates the characterization and detection of integrity issues in OWL instance data. Our work contributes the semantic web community in three aspects: (i) we provide a formal logical foundations to OWL instance data evaluation using autoepistemic operators; (ii) we implement a practical SPARQL based approach to detect the causes of integrity issues in OWL instance data as defined using ADLs; (iii) by allowing users to selectively enforce integrity constraints on individual classes and properties, we improve our previous work (that adopts global closed world assumptions) to enable more flexible local closed world reasoning.

Our future work will identify more integrity issues in Semantic Web instance data, characterize them with autoepistemic operators, and extend our SPARQL-based solutions for more expressive ontologies.

## References

1. Ding, L., Tao, J., McGuinness, D.L.: An Initial Investigation on Evaluating Semantic Web Instance Data. In: 17th International World Wide Web Conference(WWW2008), Poster, 1179-1180. (2008)
2. Donini, F.M., Bari, P.d., Nardi, D.: Description Logics of Minimal Knowledge and Negation as Failure. *ACM Trans. Comput. Logic.* 3, 177–225 (2002)
3. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A.: An Epistemic Operator for Description Logics. *J. Artif. Intell.* 100, 225–274 (1998)
4. Donini, F.M., Nardi, D., Rosati, R.: Autoepistemic Description Logics. In: 15th International Joint Conference on Artificial Intelligence (IJCAI'97). pp. 136–141. (1997)
5. Grimm, S., Motik, B.: Closed-World Reasoning in the Semantic Web through Epistemic Operators. In: 1st International Workshop on OWL Experiences and Directions (OWLED2005) (2005)
6. Lifschitz, V.: Minimal Belief and Negation as Failure. *J. Artif. Intell.* 70, 53–72 (1994)
7. Lin, F.Z., Shoham, Y.: Epistemic Semantics for Fixed-Points Non-Monotonic Logics. In: 3rd Conference on Theoretical Aspects of Reasoning about Knowledge (TARK1990),111–120, (1990)
8. Motik, B., Horrocks, I., Sattler, U.: Adding Integrity Constraints to OWL. In: 3rd International Workshop on OWL Experiences and Directions (OWLED2007) (2007)
9. Motik, B., Horrocks, I., Sattler, U.: Bridging the Gap Between OWL and Relational Databases. In: 16th international conference on World Wide Web (WWW2007), pp. 807–816 (2007)
10. Perez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. In: ISWC'06. pp. 30–43 (2006)
11. Rosati, R.: Reasoning about Minimal Belief and Negation as Failure. *J. Artif. Intell. Res.* 11, 277–300 (1999)
12. Tao, J., Ding, L., McGuinness, D.L.: Instance Data Evaluation for Semantic Web-Based Knowledge Management Systems, to appear in 42th Hawaii International Conference on System Sciences (2009)