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,dingl,baojie,dlm}@cs.rpi.edu

Abstract. We view OWL instance data evaluation as a process in which instance data is checked for conformance with application requirements. We previously 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, and a practical SPARQL-based issue checking approach that is a sound approximation for detecting integrity issues.

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

1 Introduction

Before using Semantic Web data, it is important to confirm that the instance data meets the semantic expectations of the ontology and application designers. 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 [4]. Some integrity issues depend on whether the applications include closed world assumptions (CWA) or open world assumptions (OWA). Our previous work attempts to capture integrity issues in OWL instance data by adopting global closed world assumptions. In this work, we further extend the framework to also support local closed world assumptions.

The main contributions of the paper include: (i)using Autoepistemic Description Logics (ADLs) to characterize these integrity issues as integrity constraints. This formal representation clarifies the semantics of these integrity issues which were captured by RDF graph patterns in our previous work, and enables local closed world reasoning which is more flexible than the previous global closed world reasoning; (ii)we show that issues that are characterized as ADL integrity constraints can be soundly approximated by SPARQL patterns.

2 Characterizing Integrity Issues in OWL Instance Data

Autoepistemic Description Logics (ADLs)[1] extend DLs with two modal operators **K** and **A** from nonmonotonic logic MKNF. Intuitively, **K** and **A** operators refer to minimal knowledge and assumptions respectively. An autoepistemic extension of OWL [2] has been proposed to realize closed world reasoning in the Semantic Web. In this section, we show how to model some typical integrity issues in OWL instance data using ADLs. 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.

2.1 Missing Property Value Issues (MPV)

Missing property value issues (MPV) 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`.

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

Example 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 contraint $\{\mathbf{K}\mathsf{Wine} \sqsubseteq \exists \mathbf{A}\mathsf{locatedIn}.\top\}$ is added. If the constraint is not satisfied, a MPV$\exists$ issue would occur.

Similarly, the integrity constraints corresponding to `owl:cardinality` and `owl:minCardinality` could be formalized as $IC_{MPV_{=}} = \{\mathbf{K}C \sqsubseteq (= n)\mathbf{A}P.\top\}$ and $IC_{MPV_{\geq}} = \{\mathbf{K}\mathsf{C} \sqsubseteq (\geq n)\mathbf{A}\mathsf{P}.\top\}$ respectively. The MPV$_{=}$ and MPV$_{\geq}$ issues could be defined accordingly. Here, we extend the $\mathcal{ALCK}_{NF}$ with quantification constructor $\mathcal{Q}$.

2.2 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_d Issue) *Given a knowledge base $KB = \{\mathcal{T}, \mathcal{A}\}$ which is satisfiable. Let $IC_{UIT_d} = \{\exists \boldsymbol{K}P.\top \sqsubseteq \boldsymbol{A}C\}$ for some C, P. If $\{\mathcal{T}, \mathcal{A}, IC_{UIT_d}\}$ is not satisfiable, then the ABox $\mathcal{A}$ has a UIT_d issue.*

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

Definition 4 ($UIT_{\forall}$ Issue) *Given a knowledge base $KB = \{T, A\}$ which is satisfiable. Let $IC_{UIT_{\forall}} = \{\boldsymbol{K}C \sqsubseteq \forall P.\boldsymbol{A}D\}$ for some C, D, P. If $\{T, A, IC_{UIT_{\forall}}\}$ is not satisfiable, then the ABox A has a $UIT_{\forall}$ issue.*

2.3 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 expexcted by the application.

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

We propose the extension of $\mathcal{ALCK}_{NF}$ to formalize more integrity issues, such as excessive property value issue (EPV), redundant individual issue (RIT), and uniqueness issue (UT) [5]. With this approach, users can selectively enforce integrity constraints on individual classes and properties, thus allows not only global but also local closed world reasoning with OWL instance data.

3 SPARQL-based Approximation to Integrity Issues Detection

In general, reasoning in ADLs is a hard problem. To the best of our knowledge, there is still no reasoner that can efficiently handle the reasoning in the autoepistemic extension of OWL. Our work adopts a practical SPARQL-based approach. In this section, first we give the definition of integrity violations, then we use the $MPV_{\exists}$ issue as an example to show that the SPARQL-based issue detection mechanism is a sound solution to the integrity issue detection problem. We follow the SPARQL syntax and semantics in [3].

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* $\mathcal{A}|_i = \{\alpha | \alpha \in \mathcal{A}$ *and* i *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* $i \not\simeq IC$*.*

Definition 7 ($MPV_{\exists}$ **SPARQL Pattern)** *The SPARQL pattern* $P_{MPV_{\exists}}$ *is:*

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

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* $i \not\simeq IC_{MPV_{\exists}}$*.*

Proof: The proof has two steps:

(1) 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 rdf:type C`), P_2=(`?i P ?o`), R = (`!BOUND(?o)`), Then

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

Since,

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

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 $[[P_{MPV_\exists}]]_{\mathcal{A}}$, then $C(i) \in \mathcal{A}$ 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 $i \not\simeq IC_{MPV_\exists}$, 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 $C(i) \in A|_i$ and $\nexists j$ such that $P(i,j) \in A|_i$. 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 $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \mathbf{K}C(i)$, then by $IC_{MPV_\exists}$, $(\mathcal{I}, \mathcal{M}, \mathcal{M}) \models \exists \mathbf{A}P.\top(i)$, 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'. □

The SPARQL solutions to other typical integrity issues are provided in [4]. An advantage of SPARQL-based solutions is that they are easy to implement using existing tools. We have implemented instance data evaluation as an online service (http://onto.rpi.edu/demo/oie/) which can detect typical integrity issues in instance data, in addition to syntax errors and logical inconsistencies.

4 Conclusions and Future Work

This work investigates the characterization and detection of integrity issues in OWL instance data. We provide a logical foundation to OWL instance data evaluation using autoepistemic operators, and implement a practical SPARQL-based approach to detect the issues. In future work, we will identify and characterize more integrity issues with autoepistemic operators, and extend our SPARQL-based solutions to handle more expressive ontologies.

References

1. Donini, F.M., Nardi, D., Rosati, R.: Autoepistemic Description Logics, IJCAI, pp. 136–141 (1997).
2. Grimm, S., Motik, B.: Closed-World Reasoning in the Semantic Web through Epistemic Operators, OWLED (2005).
3. Perez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL, ISWC, pp. 30–43 (2006).
4. Tao, J., Ding, L., McGuinness, D.L.: Instance Data Evaluation for Semantic Web-Based Knowledge Management Systems, to appear in HICSS, (2009).
5. Tao, J., Ding, L., Bao, J., McGuinness, D.L.: Characterizing and Detecting Integrity Issues in OWL Instance Data, TW technical report, (2008).