Towards Integrity Constraints in OWL

Evren Sirin[1] and Jiao Tao[2]

[1] Clark & Parsia, Washington, DC, USA
evren@clarkparsia.com
[2] Rensselaer Polytechnic Institute
taoj2@cs.rpi.edu

Abstract. In many data-centric applications, it is desirable to use OWL as an expressive schema language with which one expresses constraints that must be satisfied by instance data. However, specific aspects of OWL's standard semantics—i.e., the Open World Assumption (OWA) and the absence of Unique Name Assumption (UNA)—make it difficult to use OWL in this way. What triggers a constraint violation in closed world systems leads to new inferences in standard OWL systems. In this paper, we show how defining an Integrity Constraint (IC) semantics for OWL axioms can overcome this problem and discuss possible semantics for ICs. We examine IC semantics discussed in the deductive databases literature, discuss how to adopt these approaches for OWL, and compare it with existing proposals for ICs in OWL. We show IC validation problem can be reduced to SPARQL query answering using off-the-shelf reasoner and present our preliminary results with a prototype implementation using Pellet.

1 Introduction

Web Ontology Language (OWL) [13] is an expressive ontology language. The foundations of OWL are based on Description Logics (DL) for which there are sound and complete formal reasoning algorithms that can check the logical consistency of data. The semantics of OWL addresses distributed knowledge representation scenarios where complete knowledge about the domain cannot be assumed. Specifically, the following two features of OWL semantics make it suitable for such use cases:

1. OWL adopts the Open World Assumption (OWA): a statement cannot be inferred to be false on the basis of a failure to prove it.
2. OWL does *not* adopt the so-called Unique Names Assumption (UNA) which would cause OWL tools to treat two resources with different identifiers as distinct objects.

On the other hand, these features make it difficult to use OWL for data validation in applications where complete knowledge can be assumed for some or all parts of the domain. For example, for a knowledge base (KB) containing information about products in a company's inventory, it can be assumed that

manufacturer of every product is known. We would like to detect (or prevent) the case that a product is added to the KB without the manufacturer information. It is possible in OWL to express the requirement that every product has a manufacturer but due to OWA not having the manufacturer information for a product in the KB would not cause a logical inconsistency and cannot be used directly for detection.

To use OWL both as a knowledge representation language and as a constraint language for data validation, we must combine open world reasoning and closed world constraint checking. It is possible to use a different formalism like rules to express closed world constraints but this increases the cost of ontology development and maintenance as users have to deal with different formalisms. It is very desirable to use the same representation language to express axioms used for reasoning and constraints. That is, in our ontology, we would like to have the ability to turn on OWA for parts of the domain where we have incomplete knowledge and turn it off for parts where we have complete knowledge.

In this paper, we describe an alternative Integrity Constraint (IC) semantics for OWL axioms to enable closed world constraint validation. Our work is in line with recent research by Motik et al. [9] where ICs are written as standard OWL axioms but are interpreted with a different semantics for data validation purposes. We start with some use cases and example ICs (Section 2), discuss why we are not completely satisfied with existing solution (Section 3), examine how to adopt the IC approach from deductive databases research and show that with the IC semantics we propose we can reduce IC validation to SPARQL query answering (Section 4). We also present our preliminary results with a prototype IC validator implementation using Pellet (Section 5).

2 Use Cases and Motivation

There are several common use cases for closed world constraint checking that have been identified in the relational and deductive databases literature [3]. We prepared a short and non-technical user survey to gather use cases and requirements for ICs from the OWL community. The survey was published on the Web[3] and advertised on many relevant mailing lists. In addition to multiple-choice questions some survey participants provided additional uses cases and requirements for ICs. These use cases were very similar to what we consider to be the canonical IC use cases and can be summarized under the following groups:

Typing constraints Typing constraints require that individuals that participate in a relation should be instances of certain types. For example, closed world interpretation of domain and range axioms in OWL would fit into this category. In DL syntax, domain and range axioms can be expressed as follows:

$$\begin{aligned} &\exists \texttt{isManufacturedBy}.\top \sqsubseteq \texttt{Product} \\ &\top \sqsubseteq \forall \texttt{isManufacturedBy}.\texttt{Manufacturer} \end{aligned} \tag{1}$$

[3] https://clarkparsia.wufoo.com/forms/owl-integrity-constraints/

The following role assertion

$$\texttt{isManufacturedBy}(\texttt{product1}, \texttt{manufacturer1}) \tag{2}$$

would violate these ICs since individuals `product1` and `manufacturer1` are not explicitly know to be instances of `Product` and `Manufacturer` concepts respectively. The data would be valid with the addition of following assertions:

$$\begin{aligned} &\texttt{Product}(\texttt{product1}) \\ &\texttt{Manufacturer}(\texttt{manufacturer1}) \end{aligned} \tag{3}$$

Note that, when interpreted with the standard OWL semantics the assertion (2) would not trigger a logical inconsistency w.r.t the axioms of (1) and would cause the axioms of (3) to be inferred.

Domain and range axioms can be seen as global typing constraints; that is they affect instances of every class that participates in a property assertion. OWL also allows finer-grained typing constraints using `AllValuesFrom` restrictions.

Participation constraints Participation constraints require that instances of the constrained class should have a role assertion. Given an IC semantics, the `SomeValuesFrom` restrictions in OWL can be used for this purpose. For example, consider the constraint

$$\texttt{Product} \sqsubseteq \exists \texttt{isManufacturedBy}.\texttt{Manufacturer} \tag{4}$$

that requires to know the corresponding `Manufacturer` for every `Product` instance. With IC semantics, we expect the assertion

$$\texttt{Product}(\texttt{product2}) \tag{5}$$

to be invalid w.r.t. this constraint since the manufacturer is not known. The above assertion would be valid only when additional axioms in the following form are added:

$$\begin{aligned} &\texttt{isManufacturedBy}(\texttt{product2}, \texttt{manufacturer2}) \\ &\texttt{Manufacturer}(\texttt{manufacturer2}) \end{aligned} \tag{6}$$

Uniqueness constraints Uniqueness constraints require that an individual cannot participate in multiple role assertions with the same role. The keys in relational databases enforce such constraints. A similar restriction can be expressed in OWL with a `FunctionalProperty` declaration that corresponds to the following DL axiom:

$$\top \sqsubseteq \leq 1\, \texttt{isManufacturedBy} \tag{7}$$

The OWL interpretation of this axiom on the following instance data

$$\begin{aligned} &\texttt{isManufacturedBy}(\texttt{product3}, \texttt{manufacturerX}) \\ &\texttt{isManufacturedBy}(\texttt{product3}, \texttt{manufacturerY}) \end{aligned} \tag{8}$$

would result in the inference that two mentioned manufacturers are the same:

$$\texttt{manufacturerX} = \texttt{manufacturerY} \tag{9}$$

since there is no UNA and nothing contradicts the inference that these two manufacturers are same. But, with IC semantics we expect this inference not to exist and instead get an IC violation. However, we do not want to adopt UNA strictly since that would prevent us ever asserting that two different identifiers denote the same individual. We would like to retain the ability to assert equality between identifiers but we do not want the cardinality constraints in ICs infer such equalities.

3 Related Work

Existing proposals for integrating ICs into OWL typically require expressing ICs in another formalism such as rules [8] or queries [2]. An exception to this is the recent proposal by Motik et al. [9] that is motivated with same kind of use cases we explained above. Motik et al. investigates several different possible semantics for ICs in OWL and propose a semantics based on the satisfaction of ICs in minimal Herbrand models (we refer the reader to [9] for details).

Even though we find the examples presented in [9] compelling, there are several features of the proposed semantics we find to be unintuitive. In other words, the results we get with this semantics do not completely meet our expectations about ICs. For example, suppose we have the following assertions in our ontology:

$$\begin{aligned} &(\exists \texttt{isManufacturedBy}.\{\texttt{manufacturer1}, \texttt{manufacturer2}\})(\texttt{product4}) \\ &\texttt{Product}(\texttt{product4}) \\ &\texttt{Manufacturer}(\texttt{manufacturer1}) \\ &\texttt{Manufacturer}(\texttt{manufacturer2}) \end{aligned} \tag{10}$$

The manufacturer of `product4` is either `manufacturer1` or `manufacturer2` but we don't know exactly know which one. In this case, we expect the IC (4) to be violated because intuitively this IC says that the manufacturer of every product should be known. However, this IC is satisfied with the semantics of [9] because in every minimal model of the given ontology there is a manufacturer for `product4`.

Another feature we find to be unsatisfying with [9] is how the disjunctions in ICs are interpreted. For example, suppose we have the following standard OWL axioms:

$$\begin{aligned} &\texttt{Product} \sqsubseteq \texttt{Category1} \sqcup \texttt{Category2} \\ &\texttt{Product}(\texttt{product5}) \end{aligned} \tag{11}$$

The first axiom says that every product falls into one of two categories and the next assertion declares a product instance. We do not know which category `product5` belongs to but this is fine since we are using standard OWL semantics so far. Now, let's suppose we define an IC on one (and only one) of these categories as follows:

$$\texttt{Category1} \sqsubseteq \exists \texttt{categoryType}.\top \tag{12}$$

The IC is defined on a specific category and we do not know that `product5` belongs to this category. So it is reasonable to assume that the IC will not apply to `product5` and thus it will not be violated. However, there is a minimal model where `product5` is an instance of `Category1` and it does not have a `categoryType` value so this IC will be violated according to the semantics of [9].

We believe these results do not satisfy our expectations of ICs and search an alternative solution.

4 Semantics for Integrity Constraints

There has been significant amount of research to define the semantics of ICs for relational databases, deductive databases, and knowledge representation systems in general. Typically an ICs is considered to be a First Order Logic (FOL) formula. In one view [6], an IC is satisfied by a knowledge base (KB) if the KB augmented with the IC is consistent; that is there is a model of the KB where IC is true. In another competing view [7], an IC is satisfied by a KB if the KB entails the IC; that is, IC is true in all the models of the KB.

Reiter argued against both of these approaches in [11] and presented the view that ICs are epistemic in nature and are about "what the knowledge base knows". Reiter proposes that ICs should be epistemic FOL queries that will be asked to a standard KB that does not contain epistemic axioms.

We agree with Reiter in his assessment about epistemic nature of ICs and believe this is the most appropriate semantics for ICs. However, we argue that this viewpoint in not incompatible with the entailment-based semantics proposed by Lloyd and Topor [7] in the context of providing semantics for ICs in OWL. This is because Lloyd-Topor's proposal uses a non-monotonic entailment relation based on Negation-as-Failure (NAF) instead of FOL-entailment. In the following section, we investigate how we can adopt Reiter's epistemic view of ICs for defining an alternative semantics for OWL axioms and show that this approach aligns with Lloyd-Topor technique of translating FOL queries to Datalog rules.

4.1 From ICs to Epistemic Queries

Extending DLs with epistemic operator **K** has been studied [4] where **K** operator can be used in front of a concept or a role. Intuitively, $\mathbf{K}C$ represents the set of individuals that are known to be instances of C and $\mathbf{K}R$ represent the pair of individuals that are known to be related with the role R. Our first attempt to define the IC semantics for OWL axioms would be to translate an axiom such as (4) into an epistemic axiom by adding the **K** operator in front of every predicate:

$$\mathbf{K}\texttt{Product} \sqsubseteq \exists \mathbf{K}\texttt{isManufacturedBy}.\mathbf{K}\texttt{Manufacturer} \tag{13}$$

We can now define the satisfaction of ICs as the entailment of the epistemic axiom by the standard axioms in the KB. We can equivalently reduce this entailment problem to query answering using the Epistemic Query Language (EQL) [2]. EQL allows one to pose epistemic FOL queries (that contain **K** operator) against

standard FOL KBs which satisfies our requirement. Since every OWL axiom can be represented as an FOL formula we can translate (4) to the following EQL query:

$$\mathbf{K}\texttt{Product}(x) \rightarrow \exists y.(\mathbf{K}\texttt{isManufacturedBy}(x, y) \wedge \mathbf{K}\texttt{Manufacturer}(y)) \quad (14)$$

Answering this query over the KB would return the individuals that satisfy the IC. The negated query will return the individuals that violate the IC. The semantics of **K** operator in [4] and the semantics of EQL queries in [2] are given in very similar terms and it can be shown that the entailment of subsumption axiom (13) and the entailment of the epistemic query (14) is indeed equivalent.

There are couple of important points worth emphasizing regarding this approach. First, we could have turned axiom (4) into an epistemic axiom in a different way as in:

$$\mathbf{K}\texttt{Product} \sqsubseteq \mathbf{K}(\exists \texttt{isManufacturedBy}.\texttt{Manufacturer}) \quad (15)$$

The axioms (13) and (15) have very different meanings and using (15) would suffer from the problems discussed in Section 3 for the KB of (10). Furthermore, if it is really the intention of the ontology modeler to express a constraint with the meaning of (15) then it is possible to introduce a new concept name for the existential restriction in the KB

$$C = \exists \texttt{isManufacturedBy}.\texttt{Manufacturer} \quad (16)$$

and express the IC using this new name

$$\texttt{Product} \sqsubseteq C \quad (17)$$

Second, using the **K** operator for constraints limits us mostly to named individuals as pointed out in [9]. However, we do not think this is a major problem because (1) nearly all of the use cases related to ICs are based on named individuals, (2) as the previous example shows, by introducing named concepts to represent existential restrictions, we can have ICs (at least partially) interact with unnamed (inferred) individuals.

Third, due to the way we translate OWL axioms to epistemic queries, the classical negation operation used in an OWL axiom will turn into a NAF operator. This is because we simply replace every atomic concept C and every atomic role R with $\mathbf{K}C$ and $\mathbf{K}R$ respectively. So if we have $\neg C$ originally the final result will contain $\neg\mathbf{K}C$ which is an encoding of NAF (any individual not proven to be an instance of C will be an instance of $\neg\mathbf{K}C$). As a result, the meaning of the original concept is changed to have a closed world interpretation.

It is easy to verify that the semantics we get with epistemic queries satisfies the use cases we discussed in Section 2 and 3. For example, when we translate the uniqueness constraint of (7) into a negated epistemic query that will detect violations, we get the following query:

$$\mathbf{K}\texttt{isManufacturedBy}(x, y) \wedge \mathbf{K}\texttt{isManufacturedBy}(x, z) \wedge \neg\mathbf{K}(y = z) \quad (18)$$

With this query, we are asking to retrieve two values for `isManufacturedBy` property and then test if two values are equivalent (via `owl:sameAs` inferences).

This example demonstrates that we are not adopting strict UNA because x and y can bind to different individuals and the IC will not be violated as long as they are inferred to be same.

4.2 From ICs to Datalog

The IC semantics proposed by Lloyd and Topor [7] is a commonly used technique for handling integrity constraints in deductive databases [3, Chap. 11]. The idea is to express ICs as regular first order logic (FOL) queries and execute the query over a deductive database. Deductive databases cannot answer arbitrary FOL queries but the transformation rules presented by Lloyd-Topor (which is now famously known as the Lloyd-Topor transformation) can be used to convert a FOL query to a general logic program interpreted with NAF. Such program can be executed over a deductive database in a straight-forward and efficient manner yielding a practical approach for validation of ICs expressed as FOL formula.

We illustrate the Lloyd-Topor approach using the example IC (4). We start with defining a rule whose head is empty (denoted by the special symbol $\bot$) and whose body is the negation of the FOL formula representing IC (4):

$$\bot \text{ :- } \mathbf{not}\,(\texttt{Product}(x) \rightarrow \exists y.(\texttt{isManufacturedBy}(x,y) \wedge \texttt{Manufacturer}(y)))$$

If we can prove the negation of the IC from the deductive database, we infer the empty head which indicates an IC violation. The Lloyd-Topor transformation is applied to the above rule to first replace the negation of implication producing the following rule

$$\bot \text{ :- } \texttt{Product}(x) \wedge \mathbf{not}\,(\exists y.(\texttt{isManufacturedBy}(x,y) \wedge \texttt{Manufacturer}(y))) \quad (19)$$

After all transformation rules are applied, we have the following two Datalog rules as the final result

$$\begin{aligned} \bot \text{ :- } & \texttt{Product}(x) \wedge \mathbf{not}\,\texttt{P}(x,y) \\ \texttt{P}(x,y) \text{ :- } & \texttt{isManufacturedBy}(x,y) \wedge \texttt{Manufacturer}(y) \end{aligned} \quad (20)$$

where P is a new predicate generated by the transformation. It is easy to see that these transformation rules will generate a nonrecursive Datalog program.

When the Datalog program is coupled with the standard OWL axioms from the input ontology, we have a hybrid knowledge base (KB) that has an OWL component (standard ABox and TBox axioms) and an LP component (constraints). Then, detection of constraint violation can be reduced to checking if the special predicate $\bot$ is entailed by the hybrid KB. But how to check this entailment relation depends on the semantics chosen for the hybrid KB and there are several different semantics proposed in the literature to combine OWL ontologies with Datalog rules.

Given the close relationship between rules and queries[4], it is not hard to see that the epistemic query of (14) and the rules of (20) are very similar. We claim that using the semantics given for *DL-programs* in [5] will ensure that original

[4] Nonrecursive Datalog programs are known to be equivalent to unions of conjunctive queries

ontology entails the epistemic query if and only if the corresponding hybrid KB does not entail $\bot$. We only provide an informal discussion about this equivalence (formal proofs and descriptions will be included in an upcoming technical report). In DL-programs, the DL atoms in rule bodies are evaluated as queries to the OWL KB. The queries to the OWL KB contains only distinguished variables so execution of a query atom $C(x)$ (resp. $R(x, y)$) will return the same answers as $\mathbf{K}C(x)$ (resp. $\mathbf{K}R(x, y)$). The **not** operator in DL-programs is evaluated as NAF which also aligns with the interpretation of epistemic queries.

The equivalence of IC semantics for OWL axioms using Reiter's epistemic query approach and Lloyd-Topor's transformation to Datalog programs with NAF is significant for several reasons. First of all, it shows that these two approaches are not necessarily incompatible with each other. Second, we reach the same semantics starting from two different viewpoints of ICs which we take as another validation point for the IC semantics we described. Finally, the equivalence between two approaches provides different implementation possibilities.

4.3 From ICs to SPARQL queries

In the previous section, we showed how ICs can be translated to nonrecursive Datalog programs for validation. It has been shown in [1] that SPARQL [10], is the W3C's query language for RDF, has the same expressive power as nonrecursive Datalog programs. Therefore, we can also use SPARQL query answering to validate ICs. SPARQL semantics [10] is defined in terms of pattern matching on RDF graphs but the SPARQL specification allows different entailment regimes to be "plugged-in" for querying more expressive KBs. SPARQL-DL [12] defines such an entailment regime so answers to SPARQL queries will include OWL inferencing results.

The representation of epistemic queries presented in Section 4.1 (or equivalently representation of nonrecursive Datalog programs) in SPARQL is generally straight-forward but SPARQL does not directly provide an NAF operator. The well-known method for representing NAF [5] is based on a pattern of OPTIONAL/FILTER/!BOUND operators. For example, using this pattern, the SPARQL representation of IC (14) is:

```
ASK WHERE { ?x  rdf:type  Product.
            OPTIONAL {
                ?x  isManufacturedBy  ?y.
                ?y  rdf:type  Manufacturer.}
            FILTER(!BOUND(?y))}
```

In the above SPARQL query, the variable $?y$ in the filter expression should be one of the variables that appear in the OPTIONAL graph pattern but not in {`?x rdf:type Product`}. If such variable does not exist, we can add an additional triple inside the OPTIONAL graph pattern such that the variable is always

[5] `http://www.w3.org/TR/rdf-sparql-query/#func-bound`

bounded. We should also note that SPARQL WG at W3C is currently working on revising the SPARQL specification and direct representation of NAF is one of the new features that will be added to the language.[6]

5 Implementation

We have built a prototype IC validator[7] by extending Pellet[8]. The prototype includes a parser, a translator, and a validator for ICs that can read and validate ICs written as OWL, OWL 2, or SWRL axioms. The IC validator can be accessed either using a command-line program or the validation API that programmatically returns validation results.

The prototype is implemented using the SPARQL translation described in Section 4.3. Given an IC, it is translated to a SPARQL query and executed by a SPARQL engine over the Pellet reasoner. Even tough this is only a proof-of-concept implementation, it can still be used to validate relatively large datasets. For example, we took the LUBM dataset and defined five ICs over the dataset. We then used the validator to check for violations in LUBM(5) (100K instances, 800K assertions). Logical consistency checking takes 10s for this dataset whereas validating one IC takes around 2s.

The prototype validates each IC separately by executing the corresponding query. This approach would obviously not scale to thousands of ICs but there are many optimization possibilities not utilized in our implementation. It is also important to note that when an axiom is declared as an IC instead of a standard OWL axiom, this potentially reduces the expressivity of the ontology and the query answering over that ontology becomes easier. It is reasonable to expect that the expressivity of an ontology without the ICs can fit into a tractable profile (as OWL 2 QL or OWL 2 EL) which would allow efficient query answering and consequently efficient IC validation.

6 Conclusions and Future Work

In this paper, we have discussed how to define semantics for ICs in OWL. We examined IC semantics proposed in the deductive databases literature and discussed how to adopt these approaches for OWL. We showed that adopting both the epistemic-query approach of Reiter and the Lloyd-Topor translation from FOL ICs to Datalog rules with NAF give the same results when applied to ICs expressed as OWL axioms. Based on these results, we showed that IC validation can be reduced to SPARQL query answering using off-the-shelf reasoners. Our preliminary results with a prototype IC validator implementation shows promising results for efficient IC validation. Our results show that the goal of using

[6] `http://www.w3.org/TR/sparql-features/`
[7] `http://clarkparsia.com/pellet/oicv-0.1.2.zip`
[8] `http://clarkparsia.com/pellet`

OWL both as a knowledge representation and constraint language for data validation can be achieved without too much effort. We believe that adding integrity constraints in OWL greatly facilitates automated reasoning and validation in ontology-based applications.

There are several issues that will be addressed as part of our future work. For example, we did not discuss the effect of using user-defined datatypes in ICs which would require us to use built-in functions (e.g $<$ or $>$) in rules or comparable FILTER expressions in SPARQL queries. Interaction of cardinality restrictions in ICs (especially when cardinality is greater than 1) and disjunctive equality between individuals require more detailed analysis. As part of our future work we will provide a more detailed and formal investigation of these issues.

References

1. Renzo Angles and Claudio Gutierrez. The expressive power of sparql. In *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*, pages 114–129, 2008.
2. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. EQL-Lite: Effective first-order query processing in description logics. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI2007)*, pages 274–279, 2007.
3. Robert Colomb. *Deductive Databases and Their Applications*. CRC Press, 1998.
4. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1–2):225–274, 1998.
5. Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
6. Robert A. Kowalski. Logic for data description. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 77–102. Plenum Press, New York, 1978.
7. John W. Lloyd and Rodney W. Topor. Making prolog more expressive. *Journal of Logic Programming*, 1:225–240, 1984.
8. Boris Motik. A faithful integration of description logics with logic programming. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI2007)*, pages 477–482, 2007.
9. Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between owl and relational databases. In *Proceedings of the 16th international conference on World Wide Web (WWW2007)*, pages 807–816. ACM Press, 2007.
10. Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf, 2008.
11. Raymond Reiter. On integrity constraints. In *The 2nd Conference on Theoretical Aspects of Reasoning about Knowledge (TARK1988)*, pages 97–111, 1988.
12. Evren Sirin and Bijan Parsia. Sparql-dl: Sparql query for owl-dl. In *3rd OWL Experiences and Directions Workshop (OWLED-2007)*, 2007.
13. Michael K. Smith, Christopher Welty, and Deborah McGuiness. Owl web ontology language guide, 2004.