Opening, Closing Worlds — On Integrity Constraints

Evren Sirin[1], Michael Smith[1], Evan Wallace[2]

[1] Clark & Parsia LLC, Washington, DC, USA
{evren,msmith}@clarkparsia.com
[2] National Institute of Standards and Technology (NIST)
ewallace@cme.nist.gov

Abstract. In many data-centric applications it is desirable to use OWL as an expressive schema language where one expresses constraints that need to be satisfied by the (instance) data. However, some features of OWL's semantics, specifically the Open World Assumption (OWL) and not having a Unique Name Assumption (UNA), make it hard to use OWL for this task. What would trigger a constraint violation in a closed world system like a relational database leads to new inferences in OWL. In this paper, we explore how OWL can be extended to accommodate integrity constraints and discuss several alternatives for the syntax and semantics of such an extension. We primarily focus on applications in the Supply Chain Management (SCM) domain but we are also gathering use cases and requirements from many other application areas to assess which of these alternatives provides the best solution.

1 Introduction

The commercial usage of IT systems based on formal ontologies — and, in particular, ontologies formalized with Description Logics (DL) — has steadily increased since W3C completed specification of OWL. Ontologies promise, among other benefits, flexible access *to* and semantic interoperability *between* diverse, heterogeneous data sources. Ontology-based systems are, in that respect, a concrete means by which to pursue a common strategy for integrating heterogeneous systems: to build abstraction barriers between the disparate parts of those systems in order to manage the underlying, messy complexities.

But some features of OWL's semantics (where we mean, here, OWL DL, primarily) make OWL rather less useful when it comes to use cases around data validation:

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.

These features not only make it more rather than less difficult to use OWL for data validation purposes in a straightforward way; but they also consistently surprise and frustrate OWL users, particularly those who are not experts in ontology engineering. And yet, given OWL's expressiveness, using it as a flexible, abstract schema language for heterogeneous data sources is very appealing.

One compelling response to these problems described in the literature is [6]. Under funding from NIST's SBIR program, we are working on an Integrity Constraint (IC) system integrated with Pellet in response. The utility of a robust IC system for OWL is precisely that it will allow users, both expert and novice, to have their cake and to eat it too.

What do we mean? ICs make it possible to use OWL as an expressive schema language, while preserving the other core uses of OWL which rely on its standard semantics. In other words, ICs are an important consideration for OWL-based systems because they ease integration between and with data sources that adopt the Closed World Assumption (CWA) and the UNA and to which users, ontology engineers, and developers would like to apply the expressiveness of OWL in a carefully controlled way.

In the remainder of this paper, we explore the design space for an IC system. Our main objective is to provide an alternative semantics for existing OWL axioms and constructs that will make it possible to use them as ICs. We also describe possible syntax options that will indicate if an axiom should be treated as an IC or as a standard OWL axiom. We also describe some motivating use cases and requirements.

An important goal of this paper is to sanity-check our understanding of core use cases and requirements of an IC system with the OWLED community, in order to assure that it will be generally useful.

2 Motivation: Supply Chain Management

Given NIST's obligation to promote manufacturing, we focus on Supply Chain Management as a motivating use case for OWL's use as an expressive validation language. Supply Chain Management (SCM) is a species of information integration problem, in that it requires the integration of heterogeneous systems. SCM systems often employ specifications of the content of message passing systems as the primary means of accomplishing information integration. Focusing on the content of messages, which are often concretely described in terms of XML Schema or XML Infoset, provides a degree of abstraction and loose coupling between systems. But the manner in which that content is specified is often not abstract or loosely coupled enough.

Using OWL as a means of specifying message content standards in abstract models is a reasonable response to this problem, but it faces several obstacles. As we've seen above, the primary problem to directly using OWL to specify SCM message standards is also an obstacle to the adoption or commercial utility of OWL-based systems generally: it's not simple to use OWL, given its standard semantics, as an expressive schema language.

Hence, the main technical objective is to extend OWL to support an IC interpretation of OWL axioms, which requires the evaluation of OWL axioms under CWA to detect constraint violations. Constraint axioms are different from ordinary axioms because they do not infer new facts; instead, they help to detect errors and/or missing information in the data. These ICs may use any OWL construct including `HasValue`, `SomeValuesFrom`, `AllValuesFrom`, `MinCardinality`, `MaxCardinality`, and `ExactCardinality` restrictions as well as restrictions based on user defined `DataTypes` or `DataRanges`.

This requires both syntactic and semantic extensions of OWL. The primary objective for syntactic extensions is to ensure backward compatibility as much as possible. With semantic extensions, we want to ensure compatibility with OWL semantics and retain the decidability of OWL-DL.

The following are the main tasks we have identified to accomplish an initial prototype IC system:

1. Identify use cases and requirements
2. Provide semantics
3. Provide syntax
4. Implement a proof of concept tool

Identifying use cases and requirements is an important step because there are several different approaches. The decision as to which approach will be adopted should depend on the use cases and requirements. We are soliciting feedback on these requirements from OWLED community members.

3 Integrity Constraints in OWL

There are several different ways to come up with a syntax and a semantics for ICs. Different semantic options provide different results which might be considered unintuitive based on the application area. Different syntax options bring different development and maintenance cost as well as backward compatibility issues.

In the rest of this section we will briefly describe three semantics candidates studied in the literature and describe syntax alternatives for ICs. But first we will give a very simple example that will be used throughout the document to explain the differences between various semantics and syntax alternatives.

3.1 Example Integrity Constraints

Consider the following very simple OWL ontology O_1 that contains only two TBox axioms:

```
SubClassOf(Product SomeValuesFrom(madeBy Manufacturer))
SubClassOf(Product MaxCardinality(madeBy 1))
```

This ontology defines the concept of a `Product` by simply associating a product to the `Manufacturer` it is `madeBy`. The combination of two axioms ensures that each `Product` instance is related to one and only one `Manufacturer` instance. Suppose we have another ontology O_2 with the following ABox assertions:

```
ClassAssertion(prod1 Product)
ClassAssertion(prod2 Product)
PropertyAssertion(madeBy prod2 x)
ClassAssertion(prod3 Product)
PropertyAssertion(madeBy prod3 East123)
PropertyAssertion(madeBy prod3 West789)
ClassAssertion(East123 Manufacturer)
```

Although the product description in O_2 does not seem to satisfy the restrictions in O_1 there is no inconsistency according to OWL semantics. The assertion about instance `prod1` does not cause inconsistency because, under OWA, a reasoner would only assume it does not know the manufacturer of that product. The instance `prod2` does not cause inconsistency even though `x` is not known to be a `Manufacturer` instance. Missing type information would be *inferred* by the reasoner and there is no information contradicting that inference. Finally, without any more information, the assertions about `prod3` are also consistent because, without UNA, OWL semantics dictates that instances `East123` and `West789` refer to the same instance.[3]

3.2 Semantics for Integrity Constraints

We believe the most important issue to solve for adding integrity constraints to OWL is to define the semantics for constraints. OWL has a well-defined model theoretic semantics [9] which provides a normative definition of *ontology consistency*. The semantics of IC should be provided in a similarly unambiguous fashion to guarantee consistency and interoperability.

In the following sections, we explain the three main options for IC semantics that we are investigating. For each option, we provide the rationale for why we think the option should be considered and point out potential problems.

Skolemization-based Semantics (Sem1) Motik et al. [6] describe an approach to augment OWL with ICs. This approach requires only to tag some TBox axioms as being IC. These axioms are then used to check whether an ABox is of an appropriate form, but they cannot imply new ABox facts. According to the given semantics, a constraint axiom can still imply a new TBox axiom, e.g. a subclass relation can be inferred using ICs.

This semantics is based on the notion of *outer skolemization* [8] of first-order logic formulae and *minimal-model semantics*. The intuitiveness of this semantics is justified by the examples provided in the paper and with the analysis that ICs do not affect the results of ABox queries.

If the axioms in ontology O_1 from Section 3.1 were tagged as constraints with this semantics, then we would get a constraint violation for all three product instances. However, there are still unintuitive results that one can get with

[3] There would indeed be an inconsistency if we also had an additional assertion stating `DifferentIndividuals(East123 West789)`. However, in practice such distinctness assertions do not always exist.

this semantics. Specifically, TBox axioms can play an important role in the satisfaction of constraints. For example, suppose we have an ontology O_3 with the following axioms:

```
SubClassOf(ConsortiumProduct Product)
SubClassOf(ConsortiumProduct
           SomeValuesFrom(madeBy ConsortiumManufacturer))
SubClassOf(ConsortiumManufacturer Manufacturer)
EquivalentClasses(ConsortiumManufacturer
                  OneOf(East123 West789 ...))
ClassAssertion(prod4 ConsortiumProduct)
```

Unlike O_2 ontology O_3 does not violate the constraints in O_1 even though we don't know which manufacturer makes `prod4`. This is because the `SomeValuesFrom` restriction on `ConsortiumProduct` is written as a standard OWL axiom and is enough to satisfy the ICs from O_1.

There is also a compatibility issue regarding the skolemization-based semantics with existing OWL semantics. The semantics is given in terms of first-order formulae, whereas the model theoretic OWL semantics [9] is given in terms of OWL constructs. There is a clear translation between the two, but introducing first-order formulae adds a level of complexity which could make it harder for ontology modelers to understand ICs (or predict the results of constraints).

Rule-based Semantics (Sem2) ICs require some sort of CWA. One way to add CWA to OWL is through integration with logic programming (LP). LP provides negation-as-failure under CWA and thus can be used to express ICs.

In the literature, there have been many proposals to integrate DLs with LP [2, 7, 3]. The main idea is to separate the atoms into two sets — DL atoms and LP atoms — where rules on the LP side can refer to both atoms, but DL axioms can only refer to DL atoms. DL atoms are interpreted by a DL reasoner with DL semantics under OWA. LP atoms are interpreted by an LP reasoner with LP semantics under CWA.

Using LP rules to represent ICs is a simpler task than combining DLs and LP for a hybrid reasoning framework, since we are not interested in LP atoms (ICs would only refer to DL atoms) or inferring new DL atoms using LP rules. We are only interested in checking if certain conditions regarding DL atoms hold, and this can be achieved by rules that have only antecedents but no consequences. In general, deriving an empty consequence indicates an inconsistency; in this special case, it means an IC has been violated.

The two axioms from O_1 can be translated into the following LP rules:

$$\leftarrow Product(x) \wedge \textbf{not } (madeBy(x, y) \wedge Manufacturer(y))$$
$$\leftarrow Product(x) \wedge madeBy(x, y) \wedge madeBy(x, z) \wedge \textbf{not } y = z$$

Note that the DL atoms in these rules can be seen as queries to the ontology where the negation operator **not** is evaluated as negation as failure.

Query-based Semantics (Sem3) Another closely related approach for representing integrity constraints is proposed in [1]. The proposal is to use boolean epistemic queries for constraints. An epistemic query uses the *knowledge operator* **K** to indicate that query atoms should be interpreted under CWA.

For example, the following epistemic query

$$Q(x) = \mathbf{K}Product(x) \wedge \neg\mathbf{K}(madeBy(x, y) \wedge Manufacturer(y))$$

asks for all *known* `Product` instances that are *not known* to be made by a `Manufacturer`. Such a query asked against the combination of ontologies O_1 and O_2 would return both `prod1` and `prod2` since neither is an answer to the query $madeBy(x, y) \wedge Manufacturer(y)$. With this approach, the constraint axioms, as the ones in O_1, can be translated to epistemic queries that will be used to validate data.

Semantics Summary All three possibilities discussed above are closely related to each other, yet have subtle differences that might have important consequences in practice for ontology developers and tool builders. Rule-based and query-based semantics are very similar (as the above examples show) but they would differ in encoding some constructs, e.g. cardinality restrictions, and require different kinds of techniques to validate the data. Unlike skolemization-based semantics both these approaches require some kind of transformation from OWL axioms to ICs which increases the complexity of the overall solution.

3.3 Syntax

This section presents alternatives for the syntactic representation of ICs. The syntactic representation affects tools that accept ICs as input, such as validators, and those that produce them as output, such as modeling software. Although the development of a tool-specific syntax is possible, we consider that undesirable because it would act as a barrier to interoperability.

We will now present four different syntax alternatives that we are considering. We will explain the end-user flexibility each solution provide along with the cost of implementing new tools to support the serialization. We also discuss the impact of the syntax on existing OWL tools which are not explicitly extended to support ICs.

New Vocabulary (Syn1) One alternative for syntax is the creation and application of a new vocabulary for each OWL construct that can be interpreted with integrity constraint semantics. A straightforward implementation would mirror the existing OWL syntax but with a minor and consistently applied modification of necessary syntax elements indicating the alternative semantic interpretation. One such modification is to replace the OWL namespace with one specific to ICs and reuse fragment identifiers as appropriate. E.g., `owl:Restriction` is replaced by `ic:Restriction`, `owl:minCardinality` is replaced by `ic:minCardinality`, etc.

This alternative, by virtue of a distinct, separated vocabulary, has the benefit of making the presence of semantic modification obvious in the serialization.

This benefit is accompanied by significant cost. For tool maintainers choosing to add IC support to existing OWL tools, current parsers and serializers could not be reused without modification. The size of the required modification is dependent upon the existing implementation and the extent to which the new syntax mirrored the existing syntax. The possibilities range from complete reimplementation to minor changes to recognize new URIs.

The impact of a new vocabulary on OWL tools which are not updated to recognize the syntax is more significant. For tools accepting OWL input, the presence of unrecognized vocabulary would lead to parse errors, which are dealt with in a tool specific fashion. Often such errors cause abnormal termination, sometimes they are silently ignored. Regardless of actual behavior on error, it is very unlikely that any unmodified tool would be able to successfully parse the syntax in a useful fashion, and possible that the user would lose the ability to parse interleaved, non-IC axioms.

Similarly, no unmodified tools could output the syntax. Considered together, these problems prevent using existing tools for any editing or interpretation of ICs and strongly discourage adoption of a new vocabulary for ICs.

Ontology Annotation (Syn2) The simplest syntax alternative that would use OWL annotations would do so by coining a new annotation property, analogous to `owl:imports`, to identify an *ontology* which is to be interpreted in accord with IC semantics. This approach requires the ICs to be maintained in a distinct file from standard axioms.

For existing, unmodified tools processing a standard OWL ontology augmented with constraints, the behavior will be as if the IC did not exist. This happens because such tools don't recognize the new annotation property and the default semantic interpretation of annotation properties is that they have no semantic effect. For such tools, the ICs, stored in a distinct file, will be ignored.

In contrast, because (Syn2) uses existing syntax to represent ICs, a user may use existing, unmodified OWL tools for parsing, manipulation, and serialization of ICs if the IC ontology is processed independently of the target ontology (i.e., the ontology that is to be constrained by the IC axioms). Such tools will interpret the constraints using standard semantics, which may be acceptable for some tasks, such as editing, but unacceptable for others. This alternative also reduces the cost of enhancing existing tools because reusing existing OWL syntax allows existing parsing and serialization tools to be reused.

The primary deficiency of (Syn2) is that it reduces user flexibility and increases the maintenance burden by requiring that ICs be kept in a separate, distinct file from the target ontology. It may also create user confusion because the IC ontology has one set of semantics if evaluated stand-alone (that is, standard OWL semantics) and another (IC semantics) if evaluated via annotation-based inclusion from a target ontology.

Axiom Annotation (Syn3) A second syntax alternative using OWL annotations leverages the ability to annotate *axioms* rather than ontologies. Axiom annotation is included in the current OWL 2 draft specifications but was not

available in OWL 1.0 (which only allows annotation of entities, i.e., classes, properties, individuals and ontologies). By using axiom annotations, ICs could be identified using a specific annotation property and a boolean flag. For example, the following syntax samples indicate that an exact cardinality restriction should be interpreted according to IC semantics and according to standard semantics, respectively.

```
SubClassOf(
   Annotation(ic:asIntegrityConstraint "true"^^xsd:boolean)
   PurchaseOrder ExactCardinality(1 orderDate))

SubClassOf(
   Annotation(ic:asIntegrityConstraint "false"^^xsd:boolean)
   PurchaseOrder ExactCardinality(1 dueDate))
```

Note that for backward compatibility and ease of maintenance we would assume that the non-existence of an annotation means the axiom should be interpreted with the standard semantics.

Axiom annotation contrasts with ontology annotation in an obvious way: unmodified tools will process the IC axiom, but ignore the annotation indicating it is an IC. This will lead to ICs being processed in accordance with standard semantics. This may be acceptable for some tasks, such as editing, but is likely to cause problems in reasoning applications, and is the primary deficiency of the axiom annotation alternative.

However, (Syn3) allows existing parsers and serializers to be reused by tools enhanced to process ICs. Relative to use of ontology annotation, (Syn3) lowers the maintenance burden on users because ICs can be included in the same ontology as standard axioms.

A deficiency of (Syn3) is that it, like (Syn2), uses an annotation property to attain a semantic effect, despite the convention that annotation properties are presently defined in OWL 1.0 to have no semantic effects.

A risk of this alternative is its dependence on future standards activity. The behavior of legacy tools that do not support OWL 2 axiom annotations is difficult to predict because the OWL WG has not finalized the mapping of such annotations from abstract to concrete syntax, and it is currently unknown how backward compatible the mapping will be.

Rich Annotations (Syn4) The final alternative for IC syntax is based on the rich annotation proposal[4] currently being considered by the OWL WG. The proposal extends the annotation mechanism in the OWL 2 working drafts to explicitly support annotations that have semantic consequences by grouping annotations into defined *annotation spaces*. Using this approach, an annotation space would be defined for ICs and the space would have `mustUnderstand` status, which indicates that tools should not treat annotations in the space as semantics-free. IC axioms would be annotated as in the axiom annotation alternative, but each annotation would be associated with the IC annotation space.

[4] See `http://www.w3.org/2007/OWL/wiki/Annotation_System` for more details.

For the purpose of evaluating this alternative, we assume existing, unmodified tools support OWL 2 axiom annotations and rich annotations.

The primary difference between (Syn3) and (Syn4) is that tools which have not been extended to support ICs would, by virtue of the annotation space declaration, still recognize that the annotations have semantic consequence and could modify their behavior accordingly. For instance, an editor might allow the axioms to be edited and preserve the annotations, but a reasoner would likely produce an error. Thus, (Syn4) maintains the benefits of (Syn3) while minimizing its costs.

(Syn4) increases risk over (Syn3) because rich annotations are considered a more controversial feature addition to OWL 2; and, as such, may not be part of the final specification. It's notable that ICs are one of the motivating use cases put forward with the proposal to include rich annotations in the standard.

Syntax Summary (Syn1) requires a new vocabulary to be created, and all components in the toolchain to be modified. For this reason, it is considered the most disruptive alternative. (Syn2)–(Syn4) leverage OWL annotations in different ways to avoid a new vocabulary and differ primarily in the expected behavior that they would cause in existing tools, i.e. those tools which are not explicitly modified to process ICs. (Syn3) and (Syn4) depend on ongoing W3C Working Group work; selection of the most advantageous alternative will likely depend on the progression of that work.

Note that the syntax alternatives are independent of the semantics options we considered. We can translate the IC axioms written with any of the syntax options to a format required by the semantics interpretation, e.g. LP rules or epistemic queries.

3.4 Implementation Strategy

The section describes the requirements of a suite of software tools used for IC validation. Minimally, two components are anticipated in a prototype system: a syntax compiler for ICs and a data set validator.

A syntax compiler (SC) translates ICs from a persistent serialization format, e.g. RDF/XML, to an internal software representation, e.g. LP rules if a rule-based approach is adopted, that can be used by a validation tool.

A data set validator (DSV) is a software component that accepts as input a set of ICs and a data set, and which outputs, minimally, a boolean result indicating if the ICs are satisfied by the data set.

The DSV has obvious extensions beyond the minimal requirements. Most notably, in the event that the data set input does not satisfy the ICs, the component can return additional information. The simplest such extension is to report the first unsatisfied IC. Additionally, the data violating the constraint can be identified. Similarly, the validator could report not just a single violated constraint but *all* violated constraints.

Reporting causes of IC violations is related to inference explanation in Description Logic [5]. Further, existing work on presentation of precise explanations

[4] may be leveraged to improve the quality of output from the DSV. The prototype implementation will be used to explore the applicability of such techniques.

4 Conclusions

In this paper, we have presented our initial analysis of how OWL can be extended with integrity constraints. We discussed several alternatives for the syntax and the semantics of such an extension. We are seeking use cases and feedback from the OWL community to determine which alternative provides the best solution. As we reach a decision on these alternatives we are planning to implement a proof-of-concept validator as a Java software component that can be easily combined with existing OWL tools, including the Pellet reasoner and OwlSight ontology browser[5].

In this paper we have only focused on giving an IC semantics to existing OWL axioms and constructs. One obvious extension is to allow ICs that cannot be directly expressed in OWL. For example, OWL 2 allows only a restricted form of property chains (to maintain decidability) but presumably a more relaxed use of property chains in ICs would be possible. We plan to investigate such extensions as part of our future work.

References

[1] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. EQL-Lite: Effective first-order query processing in description logics. In *The 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2007)*, 2007.

[2] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: integrating datalog and description logics. *J. of Intelligent Information Systems*, 10:227–252, 1998.

[3] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. *The 9th Int'l Conf. of Knowledge Representation and Reasoning (KR2004)*, 2004.

[4] A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and E. Sirin. Beyond Axioms: Fine-Grained Justifications for Arbitrary Entailments in OWL-DL. In *International Workshop on Description Logic (DL2006)*, 2006.

[5] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *The 6th Int'l Semantic Web Conf. (ISWC2007)*, 2007.

[6] B. Motik, I. Horrocks, and U. Sattler. Bridging the Gap Between OWL and Relational Databases. In *The 16th International World Wide Web Conference (WWW2007)*, pages 807–816, Banff, AB, Canada, May 2007.

[7] B. Motik and R. Rosati. A Faithful Integration of Description Logics with Logic Programming. In *The 20th Int. Joint Conference on Artificial Intelligence (IJCAI 2007)*, Hyderabad, India, January 2007.

[8] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. 2001.

[9] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. W3C recommendation, W3C, Feb. 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.

[5] `http://pellet.owldl.com/ontology-browser/`