# Integrity Constraints in OWL

**Jiao Tao**[1], **Evren Sirin**[2], **Jie Bao**[1], **Deborah L. McGuinness**[1]
[1] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2] Clark&Parsia, Washington, DC 20001, USA

## Abstract

In many data-centric applications, we would like to use OWL to express constraints that must be satisfied by instance data. However, the Open World Assumption (OWA) in OWL's standard semantics, combined with the absence of the Unique Name Assumption (UNA), makes it difficult to use OWL in this way. What triggers constraint violations in closed world systems leads to new inferences in standard OWL systems. In this paper, we present an Integrity Constraint (IC) semantics for OWL axioms that is based on the Closed World Assumption (CWA) and the weak UNA to address this issue. Ontology modelers can choose which axioms will be interpreted with IC semantics, thus combine open world reasoning with closed world constraint validation in a flexible way. We also show that IC validation can be reduced to query answering under certain conditions. Finally, we briefly describe our prototype implementation based on the OWL reasoner Pellet.

## Introduction

The Web Ontology Language (OWL) (Smith, Welty, and McGuiness 2004; Motik, Patel-Schneider, and Grau 2009) is an expressive ontology language based on Description Logics (DL)[1] with sound and complete reasoning algorithms. The semantics of OWL addresses distributed knowledge representation scenarios where complete knowledge about the domain cannot be assumed. Further, the semantics has the following characteristics [2]:

- Open World Assumption (OWA): i.e., a statement cannot be inferred to be false on the basis of failures to prove it.
- Absence of the Unique Name Assumption (UNA): i.e., two different names may refer to the same object.

However, these characteristics can make it difficult to use OWL for data validation purposes in real-world applications where complete knowledge can be assumed for some or all parts of the domain.



[1] Throughout the paper we use the terms OWL, OWL 2, and DL interchangeably.

[2] Note that these characteristics are not just limited to OWL.

**Example 1.** *Suppose we have a KB $\mathcal{K}$ containing information about products in a company's inventory as follows:*

$\mathcal{K} = \{\texttt{Product}(p)\}$

*One might add the following axiom to express the constraint "every product is produced by a producer":*

$\alpha : \texttt{Product} \sqsubseteq \exists \texttt{hasProducer}.\texttt{Producer}$

In this example, due to the OWA, *not* having a known producer for $p$ does not cause a logical inconsistency. Therefore, we cannot use $\alpha$ to detect (or prevent) that a product is added to the KB without the producer information.

**Example 2.** *Suppose the inventory KB $\mathcal{K}$ looks like this:*

$\mathcal{K} = \{\texttt{Product}(p), \texttt{hasProducer}(p, m_1), \texttt{hasProducer}(p, m_2)\}$

*One might add the following axiom to express the constraint "a product has at most one producer":*

$\alpha : \texttt{Product} \sqsubseteq \leq 1\texttt{hasProducer}.\top$

Since $m_1$ and $m_2$ are not explicitly defined to be different from each other, they will be inferred to be same due to the cardinality restriction. However, in many cases, the reason to use functional properties is not to draw this inference, but to detect an inconsistency. When the information about instances are coming from multiple sources we cannot always assume explicit inequalities will be present.

Besides above examples, we have also identified several other requirements for ICs from the OWL community through a user survey. In these scenarios, there is a strong need to use OWL as an Integrity Constraint (IC) language with closed world semantics. That is, we would like to adopt the OWA without the UNA for parts of the domain where we have incomplete knowledge, and to use the Closed World Assumption (CWA)[3] with the UNA otherwise. This calls for the ability to combine the open world reasoning of OWL with closed world constraint validation.

In this paper, we demonstrate how to extend OWL with ICs. First, we describe an alternative IC semantics for OWL, which enables developers to augment OWL ontologies with IC axioms. Standard OWL axioms in the ontologies are used to compute inferences with open world semantics and ICs are used to validate instance data using closed world semantics. Our goal is to enable efficient data validation with

[3] With CWA, a statement is inferred to be false if it is not known to be true, which is the opposite of OWA.

OWL, especially in settings where OWL KBs are integrated with relational databases and ICs are needed to enforce the *named* individuals to have some *known* values. Then, we show that IC validation can be reduced to query answering when the KB expressivity is $\mathcal{SRI}$ or the constraint expressivity is $\mathcal{SROI}$. The queries generated from ICs can be expressed in the SPARQL query language allowing existing OWL reasoners to be used for IC validation easily.

## Related Work

The research on integrating ICs with OWL has been conducted in multiple directions. One approach to achieve this combination is to couple OWL with rule-based formalisms and express ICs as rules without heads as in (Eiter et al. 2008; Motik 2007). With this approach, ontology developers have to deal with one more additional formalism, i.e., rules, besides the ontology language OWL, to model the domain.

ICs can also be expressed with the epistemic query language EQL-Lite (Calvanese et al. 2007) where one can pose epistemic FOL queries against standard FOL KBs. Although the data complexity of answering EQL-Lite queries in DL-Lite is LOGSPACE, it would require substantially more effort to support EQL-Lite in DL KBs with full expressivity and the complexity results are still unknown.

Another line of approach is the epistemic extension of DLs (Donini et al. 1998; Donini, Nardi, and Rosati 2002) where ICs are represented as epistemic DL axioms and the satisfaction of ICs is defined as the epistemic axiom entailments. However, this approach adopts the strict UNA which is not compatible with OWL since it is possible that standard OWL axioms infer that two different names identify the same individual. While existing research has focused on epistemic extensions for relatively inexpressive $\mathcal{ALC}$ there has not been much research for combining epistemic logics with more expressive DLs.

Besides the above work, there are some other proposals concerning on integration of ICs with OWL. In this paper, we focus on approaches that reuse OWL as an IC language. Our closest related work is a proposal by Motik et al. (Motik, Horrocks, and Sattler 2007) based on a minimal Herbrand model semantics of OWL: given a OWL KB $\mathcal{K}$ consisting of inferring TBox and ABox, and a constraint TBox, an axiom in the constraint TBox is satisfied by $\mathcal{K}$ if all minimal Herbrand models satisfy it. This approach may result in counterintuitive results or a significant modeling burden in the following cases.

First, unnamed individuals can satisfy constraints, which is not desirable for closed world data validation.

**Example 3.** *Consider the KB $\mathcal{K}$ that contains a product instance p and its unknown producer, and the constraint $\alpha$ that every product has a known producer:*

$\mathcal{K} = \{A = \exists \texttt{hasProducer}.\texttt{Producer}, \texttt{Product}(p), A(p)\}$

$\alpha : \texttt{Product} \sqsubseteq A$

Since $p$ has a producer in every minimal Herbrand model of $\mathcal{K}$, $\alpha$ is satisfied, even though the producer is unknown.

Second, if a constraint needs to be satisfied only by named individuals, then a special concept $O$ has to be added into the original IC axiom, and every named individual should be asserted as an instance of $O$. This adds a significant maintenance burden on ontology developers. Although this problem can be solved by automatically introducing concept $O$ into the knowledge base and constraint axioms, the intuition behind the constraint still can not be captured.

**Example 4.** *Suppose we have a KB $\mathcal{K}$ where there are two possible producers for a product p and a constraint $\alpha$:*

$\mathcal{K} = \{B = \exists \texttt{hasProducer}.\{m_1, m_2\}, \texttt{Product}(p), B(p), \texttt{Producer}(m_1), \texttt{Producer}(m_2), O(p), O(m_1), O(m_2)\}$

$\alpha : \texttt{Product} \sqsubseteq \exists \texttt{hasProducer}.(\texttt{Producer} \sqcap O)$

The intuition behind constraint $\alpha$ is that the producer of every product should be known. Even though we do not know the producer of $p$ is $m_1$ or $m_2$ for sure, $\alpha$ is still satisfied by the semantics of (Motik, Horrocks, and Sattler 2007) because in every minimal Herbrand model $p$ has a producer that is also an instance of `Producer` and $O$.

Third, the disjunctions and ICs may interact in unexpected ways.

**Example 5.** *Consider the following KB $\mathcal{K}$ where there are two categories for products and a constraint $\alpha$ defined on one of the categories:*

$\mathcal{K} = \{\texttt{Product} \sqsubseteq \texttt{Category}_1 \sqcup \texttt{Category}_2, \texttt{Product}(p)\}$

$\alpha : \texttt{Category}_1 \sqsubseteq \exists \texttt{categoryType}.\top$

Since we are not sure that $p$ belongs to $\texttt{Category}_1$, it is reasonable to assume that the constraint will not apply to $p$ and it will not be violated. However, $\alpha$ is violated with (Motik, Horrocks, and Sattler 2007) semantics because there is a minimal model where $p$ belongs to $\texttt{Category}_1$ but it does not have a `categoryType` value.

In this paper, we present a new IC semantics for OWL that can correctly capture the intended closed world constraint semantics thus overcomes the above issues. In addition, we provide a practical query-based solution for IC validation.

## Preliminaries

### Description Logics $\mathcal{SROIQ}$

In this section, we describe the syntax and semantics of the Description Logic $\mathcal{SROIQ}$ (Horrocks, Kutz, and Sattler 2006), which is the logical underpinning of OWL 2.

Let $N_C, N_R, N_I$ be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles* and *named individuals* respectively. The $\mathcal{SROIQ}$ role $R$ is an atomic role or its inverse $R^-$. Concepts are defined inductively as follows:

$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid \geq nR.C \mid \exists R.\text{Self} \mid \{a\}$

where $A \in N_C$, $a \in N_I$, $C_{(i)}$ a concept, $R$ a role.

We use the following standard abbreviations for concept descriptions: $\bot = C \sqcap \neg C$, $\top = \neg\bot$, $C \sqcup D = \neg(\neg C \sqcap \neg D)$, $\leq nR.C = \neg(\geq n+1\, R.C)$, $\exists R.C = (\geq 1\, R.C)$, $\forall R.C = \neg(\exists R.\neg C)$, $\{a_1, \ldots, a_n\} = \{a_1\} \sqcup \cdots \sqcup \{a_n\}$.

A *$\mathcal{SROIQ}$-interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, where $\Delta$ is the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* which maps $A \in N_C$ to a subset of $\Delta$, $R \in N_R$ to a subset of $\Delta \times \Delta$, $a \in$

$N_I$ to an element of $\Delta$. The interpretation can be extended to inverse roles and complex concepts as follows:

$(R^-)^{\mathcal{I}} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}, (\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}$,

$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$,

$(\geq nR.C)^{\mathcal{I}} = \{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\} \geq n\}$

$(\exists R.\text{Self})^{\mathcal{I}} = \{x \mid \langle x, x\rangle \in R^{\mathcal{I}}\}, \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$.

where $\#$ denotes the cardinality of a set.

A $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ is a collection of TBox (terminology) and RBox (role) $\mathcal{SROIQ}$ axioms which are listed in Table 1, and Abox (assertion) axioms ($C(a)$, $R(a, b)$, $a = b$, $a \neq b$) wehre their semantics is given by encoding them as TBox axioms ($\{a\} \sqsubseteq C$, $\{a\} \sqsubseteq \exists R.\{b\}$, $\{a\} \sqsubseteq \{b\}$, $\{a\} \sqsubseteq \neg\{b\}$, resp.).

| Type | Axiom | Condition on $\mathcal{I}$ |
|---|---|---|
| TBox | $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| RBox | $R_1 \sqsubseteq R_2$ | $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$ |
| | $R_1 \ldots R_n \sqsubseteq R$ | $R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$ |
| | $\texttt{Ref}(R)$ | $\forall x \in \Delta : \langle x, x\rangle \in R^{\mathcal{I}}$ |
| | $\texttt{Irr}(R)$ | $\forall x \in \Delta : \langle x, x\rangle \notin R^{\mathcal{I}}$ |
| | $\texttt{Dis}(R_1, R_2)$ | $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$ |

Table 1: Axiom satisfactions in $\mathcal{SROIQ}$-interpretation

We say that an interpretation $\mathcal{I}$ *satisfies* a $\mathcal{SROIQ}$ axiom $\alpha$, denoted $\mathcal{I} \models \alpha$ if the corresponding condition on $\mathcal{I}$ in Table 1 is satisfied. $\mathcal{I}$ is a *model* of $\mathcal{K}$ if it satisfies all the axioms in $\mathcal{K}$. We define $Mod(\mathcal{K})$ to be the set of all interpretations that are models of $\mathcal{K}$. We say $\mathcal{K}$ *entails* $\alpha$, written as $\mathcal{K} \models \alpha$, if $\mathcal{I} \models \alpha$ for all models $\mathcal{I} \in Mod(\mathcal{K})$.

## Distinguished Conjunctive Query (DCQ)

We now describe the syntax and semantics of *distinguished conjunctive query* (DCQ). Let $N_V$ be a non-empty set of variable names disjoint from $N_I$, $N_C$, and $N_R$. A *query atom* is an ABox axiom where variables can be used in place of individuals. Formally, it is defined as follows:

$q \leftarrow C(x) \mid R(x, y) \mid \neg R(x, y) \mid x = y \mid x \neq y$

where $x, y \in N_I \cup N_V$, $C$ is a concept, and $R$ is a role. A *conjunctive query (CQ)* is a conjunction of query atoms:

$Q \leftarrow q \mid Q_1 \wedge Q_2$

A DCQ is a CQ containing only distinguished variables.[4]

The semantics of DCQ is given in terms of interpretations defined in previous subsection We define an *assignment* $\sigma : N_V \rightarrow N_I$ to be a mapping from the variables used in the query to named individuals in the KB. We define $\sigma(Q)$ to denote the application of an assignment $\sigma$ to a query $Q$ such that the variables in the query are replaced with individuals according to the mapping. We say a KB $\mathcal{K}$ entails a $DCQ$ $Q$ with an assignment $\sigma$, written as $\mathcal{K} \models^{\sigma} Q$, if:

$\mathcal{K} \models^{\sigma} q$ iff $\mathcal{K} \models \sigma(q)$

$\mathcal{K} \models^{\sigma} Q_1 \wedge Q_2$ iff $\mathcal{K} \models^{\sigma} Q_1$ and $\mathcal{K} \models^{\sigma} Q_2$

We define the *answers to a query*, $\mathbf{A}(Q, \mathcal{K})$, to be the set of all assignments for which the KB entails the query. That is, $\mathbf{A}(Q, \mathcal{K}) = \{\sigma \mid \mathcal{K} \models^{\sigma} Q\}$. We say that a query is true w.r.t. a KB, denoted $\mathcal{K} \models Q$, if there is at least one answer for the query, and false otherwise.

[4] A distinguished variable can be mapped to only known individuals, i.e., an element from $N_I$

# IC Semantics

There has been a significant amount of research to define the semantics of ICs for relational databases, deductive databases, and knowledge representation systems in general. There are several proposals based on KB consistency or KB entailment. Against both of these approaches, Reiter argued that ICs are epistemic in nature and are about "what the knowledge base knows" in (Reiter 1988). He proposed that ICs should be epistemic first-order queries that will be asked to a standard KB that does not contain epistemic axioms.

We agree with Reiter in his assessment about the epistemic nature of ICs and believe this is the most appropriate semantics for ICs. In what follows, we first describe an alternative IC semantics for OWL axioms, which is similar to how the semantics of epistemic DL $\mathcal{ALCK}$ (Donini et al. 1998) and MKNF DL $\mathcal{ALCK}_{\mathcal{NF}}$ (Donini, Nardi, and Rosati 2002) are defined. Then, we discuss how the IC semantics addresses the issues explained in introduction and related work sections, and enables OWL to be an IC language.

## Formalization

We define *IC-interpretation* as a pair $\mathcal{I}, \mathcal{U}$ where $\mathcal{I}$ is a $\mathcal{SROIQ}$ interpretation defined over the domain $\Delta^{\mathcal{I}}$ and $\mathcal{U}$ is a set of $\mathcal{SROIQ}$ interpretations. The IC-interpretation function $.^{\mathcal{I},\mathcal{U}}$ maps concepts to a subset of $\Delta$, roles to a subset of $\Delta \times \Delta$ and individuals to an element of $\Delta$ as follows:

$C^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \forall \mathcal{J} \in \mathcal{U}, x^{\mathcal{J}} \in C^{\mathcal{J}}\}$

$R^{\mathcal{I},\mathcal{U}} = \{\langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \mid x, y \in N_I \text{ s.t. } \forall \mathcal{J} \in \mathcal{U}, \langle x^{\mathcal{J}}, y^{\mathcal{J}}\rangle \in R^{\mathcal{J}}\}$

where $C$ is an atomic concept and $R$ is a role. According to this definition, $C^{\mathcal{I},\mathcal{U}}$ is the interpretation of named individuals that are instances of $C$ in every (conventional) interpretation from $\mathcal{U}$. $R^{\mathcal{I},\mathcal{U}}$ can be understood similarly.

IC-interpretation $\mathcal{I}, \mathcal{U}$ is extended to inverse roles and complex concepts as follows:

$(R^-)^{\mathcal{I},\mathcal{U}} = \{\langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \mid \langle y^{\mathcal{I}}, x^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}}\}$,

$(C \sqcap D)^{\mathcal{I},\mathcal{U}} = C^{\mathcal{I},\mathcal{U}} \cap D^{\mathcal{I},\mathcal{U}}$, $(\neg C)^{\mathcal{I},\mathcal{U}} = N_I \setminus C^{\mathcal{I},\mathcal{U}}$,

$(\geq nR.C)^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \#\{y^{\mathcal{I}} \mid \langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}} \text{ and } y^{\mathcal{I}} \in C^{\mathcal{I},\mathcal{U}}\} \geq n\}$,

$(\exists R.\text{Self})^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \langle x^{\mathcal{I}}, x^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}}\}$,

$\{a\}^{\mathcal{I},\mathcal{U}} = \{a^{\mathcal{I}}\}$.

We can see that the IC-interpretation $\mathcal{I}, \mathcal{U}$ is using the closed-world assumption. For example, the elements of $C^{\mathcal{I},\mathcal{U}}$ are the interpretation of named individuals that should be in the interpretation set of $C^{\mathcal{I}}$ for all $\mathcal{I} \in \mathcal{U}$. Any named individual that can not be proven to be an instance of $C$ is assumed to be an instance of $\neg C$ since $(\neg C)^{\mathcal{I},\mathcal{U}}$ is the complement of $C^{\mathcal{I},\mathcal{U}}$ w.r.t. $N_I$.

Note that, although the IC interpretations have some similarities to the epistemic interpretations of $\mathcal{ALCK}$ and

| Type | Axiom | Condition on $\mathcal{I},\mathcal{U}$ |
|---|---|---|
| TBox | $C \sqsubseteq D$ | $C^{\mathcal{I},\mathcal{U}} \subseteq D^{\mathcal{I},\mathcal{U}}$ |
| RBox | $R_1 \sqsubseteq R_2$ | $R_1^{\mathcal{I},\mathcal{U}} \subseteq R_2^{\mathcal{I},\mathcal{U}}$ |
| | $R_1 \dots R_n \sqsubseteq R$ | $R_1^{\mathcal{I},\mathcal{U}} \circ \dots \circ R_n^{\mathcal{I},\mathcal{U}} \subseteq R^{\mathcal{I},\mathcal{U}}$ |
| | $\texttt{Ref}(R)$ | $\forall x \in N_I : \langle x^{\mathcal{I},\mathcal{U}}, x^{\mathcal{I},\mathcal{U}} \rangle \in R^{\mathcal{I},\mathcal{U}}$ |
| | $\texttt{Irr}(R)$ | $\forall x \in N_I : \langle x^{\mathcal{I},\mathcal{U}}, x^{\mathcal{I},\mathcal{U}} \rangle \notin R^{\mathcal{I},\mathcal{U}}$ |
| | $\texttt{Dis}(R_1, R_2)$ | $R_1^{\mathcal{I},\mathcal{U}} \cap R_2^{\mathcal{I},\mathcal{U}} = \emptyset$ |

Table 2: Axiom satisfactions in IC-interpretation

$\mathcal{ALCK}_{\mathcal{NF}}$ (Donini et al. 1998; Donini, Nardi, and Rosati 2002), there are some important differences. First, the IC interpretation in our approach is applicable to any $\mathcal{SROIQ}$ DL KB while the expressivity of DLs in (Donini et al. 1998; Donini, Nardi, and Rosati 2002) is limited to $\mathcal{ALC}$. Second, in $\mathcal{ALCK}$ and $\mathcal{ALCK}_{\mathcal{NF}}$ (Donini et al. 1998; Donini, Nardi, and Rosati 2002), strict UNA is used by the interpretations which is not the case in IC interpretations.

In our IC semantics, we want to adopt a weak form of UNA; that is, two named individuals with different identifiers are assumed to be different by default unless their equality is required to satisfy the axioms in the KB. This idea is similar to minimal model semantics where equality relation is treated as a congruence relation and minimized.

We formalize this notion of weak UNA by defining Minimal Equality (ME) models. We start by defining the $\prec_=$ relation. Given two models $\mathcal{I}$ and $\mathcal{J}$, we say $\mathcal{J} \prec_= \mathcal{I}$ if all of the following conditions hold:

- For every concept C, $\mathcal{J} \models C(a)$ implies $\mathcal{I} \models C(a)$;
- For every role R, $\mathcal{J} \models R(a,b)$ implies $\mathcal{I} \models R(a,b)$;
- $E_{\mathcal{J}} \subset E_{\mathcal{I}}$

where $E_{\mathcal{I}}$ is the set of equality relations between named individuals (equality relations, for short) satisfied by $\mathcal{I}$:

$E_{\mathcal{I}} = \{\langle a,b \rangle \mid a,b \in N_I \text{ s.t. } \mathcal{I} \models a = b\}$

$Mod_{ME}(\mathcal{K})$ is the models of $\mathcal{K}$ with minimal equality (ME) between named individuals. Formally, we define

$$Mod_{ME}(\mathcal{K}) = \{\mathcal{I} \in Mod(\mathcal{K}) \mid \nexists \mathcal{J}, \mathcal{J} \in Mod(\mathcal{K}), \mathcal{J} \prec_= \mathcal{I}\}$$

It is easy to see that for every ME model $\mathcal{I}$ in $Mod_{ME}(\mathcal{K})$, there is no model $\mathcal{J}$ of $\mathcal{K}$ where $E_{\mathcal{J}} \subset E_{\mathcal{I}}$. Two different named individuals are interpreted as equivalent in $\mathcal{I} \in Mod_{ME}(\mathcal{K})$ only if this equality is necessary to make $\mathcal{I}$ being a model of $\mathcal{K}$. For example, suppose we have the axiom $a = \{b\} \sqcup \{c\}$ in $\mathcal{K}$. Then, $\forall \mathcal{I} \in Mod(\mathcal{K})$, one of the following three conditions hold: (1) $a^{\mathcal{I}} = b^{\mathcal{I}}, a^{\mathcal{I}} \neq c^{\mathcal{I}}$; (2) $a^{\mathcal{I}} = c^{\mathcal{I}}, a^{\mathcal{I}} \neq b^{\mathcal{I}}$; (3) $a^{\mathcal{I}} = b^{\mathcal{I}} = c^{\mathcal{I}}$. If (1) or (2) holds, then $\mathcal{I} \in Mod_{ME}(\mathcal{K})$ because $a$ has to be interpreted to be equivalent to at least one of $b$ and $c$ to make $\mathcal{I}$ being a model of $\mathcal{K}$. Whereas for case (3), $\mathcal{I} \notin Mod_{ME}(\mathcal{K})$ since the equality relations in $\mathcal{I}$ are not minimal.

The satisfaction of axiom $\alpha$ in an IC-interpretation $\mathcal{I},\mathcal{U}$, denoted as $\mathcal{I},\mathcal{U} \models \alpha$, can be defined analogously as in conventional interpretations. We give the definitions in Table 2 for completeness. Given a $\mathcal{SROIQ}$ KB $\mathcal{K}$ and a $\mathcal{SROIQ}$ constraint $\alpha$, the IC-satisfaction of $\alpha$ by $\mathcal{K}$, i.e., $\mathcal{K} \models_{IC} \alpha$, is defined as:

$\mathcal{K} \models_{IC} \alpha$ iff $\forall \mathcal{I} \in \mathcal{U}, \mathcal{I}, \mathcal{U} \models \alpha$, where $\mathcal{U} = Mod_{ME}(\mathcal{K})$

We define an *extended KB* as a pair $\langle \mathcal{K}, \mathcal{C} \rangle$ where $\mathcal{K}$ is a $\mathcal{SROIQ}$ KB interpreted with standard semantics and $\mathcal{C}$ is a set of $\mathcal{SROIQ}$ axioms interpreted with constraint semantics. We say that $\langle \mathcal{K}, \mathcal{C} \rangle$ is valid if $\forall \alpha \in \mathcal{C}, \mathcal{K} \models_{IC} \alpha$, otherwise there is an IC violation.

## Discussion

It is easy to verify that the IC semantics provides expected results for the examples presented in introduction and related work sections, and enables OWL to be an IC language. In Example 1, we get an IC violation since the IC interpretation of $\texttt{Product}$ contains $p$ but the IC interpretation of $(\exists \texttt{hasProducer}.\texttt{Producer})$ is empty. We also get an IC violation for Example 2 because due to the weak UNA $m_1$ and $m_2$ are interpreted as different individuals causing the IC interpretation of $(\leq 1 \texttt{hasProducer}.\top)$ to be empty. In Example 3 and Example 4 there are also IC violations because the constraint requires the same named producer to exist in every ME model of the KB which is not the case. Since our IC semantics is targeted at named individuals, one does not need to use the concept $O$ as in Example 4. In Example 5, the IC interpretation of $\texttt{Category}_1$ is empty, therefore the constraint does not apply to $p$ and there is no violation.

The following example shows how weak UNA allows the individuals that are not asserted to be equal to be treated different for constraint validation purposes.

**Example 6.** *Consider the KB $\mathcal{K}$ and the constraint $\alpha$:*

$\mathcal{K} = \{C(c), R(c,d_1), R(c,d_2), D(d_1), D(d_2)\}$

$\alpha : C \sqsubseteq \geq 2R.D$

Individuals $d_1$ and $d_2$ are interpreted to be different in every ME model. Therefore, the IC-interpretation of $(\geq 2R.D)$ includes c and the constraint $\alpha$ is satisfied by $\mathcal{K}$.

Now we illustrate another point regarding disjunctions in constraints.

**Example 7.** *Suppose we have the KB $\mathcal{K}$ and constraint $\alpha$:*

$\mathcal{K} = \{C(a), (C_1 \sqcup C_2)(a)\}$

$\alpha : \quad C \sqsubseteq C_1 \sqcup C_2$

Constraint $\alpha$ should be read as "every instance of $C$ should be either a known instance of $C_1$ or a known instance of $C_2$". Since we do not know *for sure* whether $a$ belongs to $C_1$ or $C_2$, $\alpha$ is expected to be violated by $\mathcal{K}$. Indeed, according to our semantics we get $C^{\mathcal{I},\mathcal{U}} = \{a^{\mathcal{I}}\}$ and $(C_1 \sqcup C_2)^{\mathcal{I},\mathcal{U}} = \emptyset$. Therefore $C^{\mathcal{I},\mathcal{U}} \not\subseteq (C_1 \sqcup C_2)^{\mathcal{I},\mathcal{U}}$ and we conclude there is an IC violation.

If we want to represent the alternative constraint: "every instance of C should be an instance of $C_1$ or $C_2$", we can define a new name $C'$ in the KB to substitute $C_1 \sqcup C_2$, thus having the new KB $\mathcal{K}'$ and constraint $\alpha'$ as follows:

$\mathcal{K}' = \{C(a), (C_1 \sqcup C_2)(a), C' \equiv C_1 \sqcup C_2\}$

$\alpha' : C \sqsubseteq C'$

There is no IC violation in this version because now the disjunction is interpreted as standard OWL axioms. As these examples show, we can model the constraints to express different disjunctions in a flexible way.

## IC Validation

We have defined in previous section that, the extended KB $\langle \mathcal{K}, \mathcal{C} \rangle$ is valid if for every IC axiom in $\mathcal{C}$ is IC-satisfied by $\mathcal{K}$. In this section, we describe how to do IC validation, i.e., check IC-satisfaction by translating constraint axioms to queries with the *Negation As Failure (NAF)* operator "**not**". We start by giving the formal semantics for DCQ$^{\textbf{not}}$, then describe the translation rules from IC axioms to DCQ$^{\textbf{not}}$, and finally provide a theorem showing that IC validation can be reduced to answering DCQ$^{\textbf{not}}$ under certain conditions.

### DCQ$^{\textbf{not}}$

In the section of preliminaries, we introduced standard DCQ. However, the expressivity of standard DCQ is not enough to capture the closed world nature of IC semantics. For this reason, we add the **not** operator to DCQ to get DCQ$^{\textbf{not}}$. The syntax of DCQ$^{\textbf{not}}$ is defined as follows:

$Q \leftarrow q \mid Q_1 \wedge Q_2 \mid \textbf{not}\, Q$

The semantics of **not** is defined as:

$\mathcal{K} \models^{\sigma} \textbf{not}\, Q \quad \text{iff} \quad \nexists \sigma' \text{ s.t. } \mathcal{K} \models^{\sigma'} \sigma(Q)$

Recall that, assignment functions $\sigma$ map variables only to named individuals so a brute force way to check if a **not** atom is entailed can be done by enumerating all possible assignments and checking for entailment.

### Translation Rules: from ICs to DCQ$^{\textbf{not}}$

We now present the translation rules from IC axioms to DCQ$^{\textbf{not}}$. The translation rules are similar in the spirit to the Lloyd-Topor transformation (Lloyd 1987) but instead of rules we generate DCQ$^{\textbf{not}}$. The idea behind the translation is to translate a constraint axiom into a query such that when the constraint is violated the KB entails the query. In other words, whenever the answer set of the query is not empty, we can conclude that the constraint is violated.

The translation contains two operators: $\mathcal{T}_c$ for translating concepts and $\mathcal{T}$ for translating axioms. $\mathcal{T}_c$ is a function that takes a concept expression and a variable as input and returns a DCQ$^{\textbf{not}}$ as the result:

$\mathcal{T}_c(C_a, x) := C_a(x)$

$\mathcal{T}_c(\neg C, x) := \textbf{not}\, \mathcal{T}_c(C, x)$

$\mathcal{T}_c(C_1 \sqcap C_2, x) := \mathcal{T}_c(C_1, x) \wedge \mathcal{T}_c(C_2, x)$

$\mathcal{T}_c(\geq nR.C, x) :=$

$$\bigwedge_{1 \leq i \leq n} (R(x, y_i) \wedge \mathcal{T}_c(C, y_i)) \bigwedge_{1 \leq i < j \leq n} \textbf{not}\, (y_i = y_j)$$

$\mathcal{T}_c(\exists R.\text{Self}, x) := R(x, x)$

$\mathcal{T}_c(\{a\}, x) := (x = a)$

where $C_a$ is an atomic concept, $C_{(i)}$ is a concept, $R$ is a role, $a$ is an individual, $x$ is a variable, and $y_{(i)}$ is a fresh variable.

$\mathcal{T}$ is a function that maps a $\mathcal{SROIQ}$ axiom to a DCQ$^{\textbf{not}}$ as follows:

$\mathcal{T}(C_1 \sqsubseteq C_2) := \mathcal{T}_c(C_1, x) \wedge \textbf{not}\, \mathcal{T}_c(C_2, x)$

$\mathcal{T}(R_1 \sqsubseteq R_2) := R_1(x, y) \wedge \textbf{not}\, R_2(x, y)$

$\mathcal{T}(R_1 \ldots R_n \sqsubseteq R) :=$

$R_1(x, y_1) \wedge \ldots \wedge R_n(y_{n-1}, y_n) \wedge \textbf{not}\, R(x, y_n)$

$\mathcal{T}(\texttt{Ref}(R)) := \textbf{not}\, R(x, x)$

$\mathcal{T}(\texttt{Irr}(R)) := R(x, x)$

$\mathcal{T}(\texttt{Dis}(R_1, R_2)) := R_1(x, y) \wedge R_2(x, y)$

where $C_{(i)}$ is a concept, $R_{(i)}$ is a role, $x$ and $y_{(i)}$ are variables.

**Example 8.** *Given the constraint $\alpha$ in Example 1, applying the above translation rules, we get:*

$\mathcal{T}(\alpha) := \texttt{Product}(x) \wedge$

$\quad \textbf{not}\, (\texttt{hasProducer}(x, y) \wedge \texttt{Producer}(y))$

### Reducing IC Validation to Answering DCQ$^{\textbf{not}}$

We have discussed in previous subsection, we want to reduce the problem of IC validation to query answering. However, the following example shows that when both the KB and the constraints use full expressivity of $\mathcal{SROIQ}$, we cannot use the query translation approach in a straightforward way.

**Example 9.** *Suppose we have a KB $\mathcal{K}$ and a constraint $\alpha$:*

$\mathcal{K} = \{D(d), R(d, a), R(d, b), R(d, c), \{a\} \sqsubseteq \{b, c\}\}$

$\alpha : D \sqsubseteq \leq 2R.\top$

In all models of $\mathcal{K}$, $a$ is either interpreted to be equivalent to $b$ or $c$. Therefore, in all interpretations $d$ has less than two $R$ values satisfying the constraint. However, the query translation will yield atoms in the form **not** $(y_1 = y_2)$ which will be true for any individual pair in this KB. As a result, the answer set for this query will include $d$ which incorrectly indicates an IC violation. This is because of the interaction between disjunctive (in)equality axioms in $\mathcal{K}$ and cardinality restrictions in ICs: axiom $(\{a\} \sqsubseteq \{b, c\})$ is asserting disjunctive equality between individuals which result in a situation where the IC axiom $\alpha$ containing cardinality restrictions is satisfied in different ways at different interpretations. This is a very challenging task to tackle so we focus on cases where there will not be such interactions.

We can limit this interaction by either prohibiting cardinality restrictions in ICs or by prohibiting disjunctive (in)equality to appear in KBs. In $\mathcal{SROIQ}$, there are only three ways to infer (in)equality between individuals: using (1) explicit (in)equality axioms; (2) nominals (as seen above); and (3) cardinality restrictions. Obviously, explicit ABox assertions cannot be disjunctive so they are not problematic. By excluding nominals and cardinality restrictions from $\mathcal{SROIQ}$, we get the DL $\mathcal{SRI}$.

In Theorem 1, we show that IC validation via query answering is sound and complete for the cases where the expressivity of the extended KB is either $\langle \mathcal{SRI}, \mathcal{SROIQ} \rangle$ or $\langle \mathcal{SROIQ}, \mathcal{SROI} \rangle$. Due to space limitations we only present the main theorem here and refer the reader to the technical report (Tao et al. 2010) for details.

**Theorem 1.** *Given an extended KB $\langle \mathcal{K}, \mathcal{C} \rangle$ with expressivity $\langle \mathcal{SRI}, \mathcal{SROIQ} \rangle$ ($\langle \mathcal{SROIQ}, \mathcal{SROI} \rangle$ resp.), we say that $\mathcal{K} \models_{IC} \alpha$ iff $\mathcal{K} \not\models \mathcal{T}(\alpha)$ where $\alpha \in \mathcal{C}$.*

## Implementation

The most widely used query language on the Semantic Web is SPARQL (Prud'hommeaux and Seaborne 2008) which allows querying over OWL ontologies via OWL entailment

regimes. It is known that SPARQL has the same expressive power as nonrecursive Datalog programs (Angles and Gutierrez 2008) and can express DCQ$^{\mathbf{not}}$. Therefore, based on the results from previous section, we can reduce IC validation to SPARQL query answering if the KB is $\mathcal{SRI}$ or the ICs do not contain cardinality restrictions.

In Table 3, we present a mapping $\mathcal{M} : Q \rightarrow P$ from a DCQ$^{\mathbf{not}}$ $Q$ to a SPARQL graph pattern $P$, such that $\mathbf{A}(Q, \mathcal{K})$ will be equivalent to the result set obtained by evaluating $P = \mathcal{M}(Q)$ over the corresponding OWL ontology using OWL entailment regime. Note that, in our mapping we use NOT EXISTS pattern which is being added in SPARQL 1.1[5] and can also be encoded in SPARQL 1.0 using a combination of other operators (Angles and Gutierrez 2008).

| **Q** | **P** = $\mathcal{M}$(**Q**) |
|---|---|
| $C(x)$ | `x rdf:type C` |
| $R(x, y)$ | `x R y` |
| $x = y$ | `x owl:sameAs y` |
| $x \neq y$ | `x owl:differentFrom y` |
| $Q_1 \wedge Q_2$ | JOIN($\mathcal{M}(Q_1), \mathcal{M}(Q_2)$) |
| **not** $Q$ | NOT EXISTS($\mathcal{M}(Q)$) |

Table 3: Mapping DCQ$^{\mathbf{not}}$ to SPARQL queries

Based on the above mapping, the SPARQL representation of the query in Example 8 becomes:

```
ASK WHERE {
   ?x rdf:type Product.
   NOT EXISTS{?x hasProducer ?y.
              ?y rdf:type Producer.}}
```

We have built a prototype IC validator[6] by extending the OWL 2 DL reasoner Pellet[7]. The prototype reads OWL IC axioms, translates each IC first to a DCQ$^{\mathbf{not}}$ and then to a SPARQL query which is executed by the SPARQL engine in Pellet where a non-empty result indicates a constraint violation. Since the translation algorithm is reasoner independent this prototype can be used in conjunction with any OWL reasoner that supports SPARQL query answering.

We have used this prototype to validate ICs with several large ontologies such as the LUBM dataset. For testing, we removed several axioms from the LUBM ontology and declared them as ICs instead. The dataset is logically consistent but turning axioms into ICs caused some violations to be detected. Since each constraint is turned into a separate query there is no dependence between the validation time of different constraints. We have not performed extensive performance analysis for IC validation but as a simple comparison we looked at logical consistency checking time vs. IC validation time. For LUBM(5), which has 100K individuals and 800K ABox axioms, logical consistency checking was in average 10s whereas validating a single IC took in average 2s. The naive approach in our prototype to execute each query separately would not scale well as the number of ICs increase. However, there are many improvement possibilities ranging from combining similar queries into a single query to running multiple queries in parallel.

[5]http://www.w3.org/TR/sparql11-query/#negation

[6]http://clarkparsia.com/pellet/oicv-0.1.2.zip

[7]http://clarkparsia.com/pellet

## Conclusions and Future Work

In this paper, we described how to provide an IC semantics for OWL axioms that can be used for data validation purposes. Our IC semantics provide intuitive results for various different use cases we examined. We presented translation rules from IC axioms to DCQ$^{\mathbf{not}}$, showing that IC validation can be reduced to query answering when the KB expressivity is $\mathcal{SRI}$ or constraint expressivity is $\mathcal{SROI}$. Our preliminary results with a prototype IC validator implementation show that existing OWL reasoners can be used for IC validation efficiently with little effort. Using SPARQL queries for IC validation makes our approach applicable to a wide range of reasoners. In the future, we will be looking at the performance of IC validation in realistic datasets and will be exploring the IC validation algorithms for the full expressivity of $\mathcal{SROIQ}$.

## References


Angles, R., and Gutierrez, C. 2008. The expressive power of sparql. In *Proc. ISWC2008*, 114–129.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. EQL-Lite: Effective first-order query processing in description logics. In *Proc. IJCAI2007*, 274–279.

Donini, F. M.; Lenzerini, M.; Nardi, D.; Nutt, W.; and Schaerf, A. 1998. An epistemic operator for description logics. *AI* 100(1–2):225–274.

Donini, F. M.; Nardi, D.; and Rosati, R. 2002. Description logics of minimal knowledge and negation as failure. *ACM Trans. Comput. Logic* 3(2):177–225.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *AI* 172(12-13):1495–1539.

Horrocks, I.; Kutz, O.; and Sattler, U. 2006. The even more irresistible sroiq. In *Proc. KR2006*, 57–67.

Lloyd, J. W. 1987. *Foundations of Logic Programming*.

Motik, B.; Horrocks, I.; and Sattler, U. 2007. Bridging the gap between owl and relational databases. In *Proc. WWW2007*, 807–816.

Motik, B.; Patel-Schneider, P. F.; and Grau, B. C. 2009. Owl 2 web ontology language direct semantics.

Motik, B. 2007. A faithful integration of description logics with logic programming. In *Proc. IJCAI2007*, 477–482.

Prud'hommeaux, E., and Seaborne, A. 2008. Sparql query language for rdf.

Reiter, R. 1988. On integrity constraints. In *Proc. TARK1988*, 97–111.

Smith, M. K.; Welty, C.; and McGuiness, D. 2004. Owl web ontology language guide.

Tao, J.; Sirin, E.; Bao, J.; and McGuinness, D. 2010. Integrity constraints in owl. Technical report, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY.