Proc. 23rd Int. Workshop on Description Logics (DL2010), CEUR-WS 573, Waterloo, Canada, 2010.

Extending OWL with Integrity Constraints

Jiao Tao[1], Evren Sirin[2], Jie Bao[1], and Deborah L. McGuinness[1]

[1] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA
[2] Clark&Parsia, LLC, Washington, DC, USA

1 Introduction

The Web Ontology Language (OWL) [1] is an expressive ontology language based on Description Logics (DL)[1]. 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:

- 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.

Example 1 *Suppose we have the following inventory KB $\mathcal{K}$. One might add the following axiom α to express the constraint "a product is produced by a producer".*
$\mathcal{K} = \{\texttt{Product}(p)\}, \quad \alpha : \texttt{Product} \sqsubseteq \exists \texttt{hasProducer}.\texttt{Producer}$

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

Example 2 *Suppose we have the following inventory KB $\mathcal{K}$. One might add the following axiom α to express the constraint "a product has at most one producer".*
$\mathcal{K} = \{\texttt{Product}(p), \texttt{hasProducer}(p, m_1), \texttt{hasProducer}(p, m_2)\}$,
$\alpha : \texttt{Product} \sqsubseteq \leq 1\texttt{hasProducer}.\top$

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.

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

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

knowledge, and the Closed World Assumption (CWA)[2] with UNA otherwise. This calls for the ability to combine the open world reasoning of OWL with closed world constraint validation.

In this paper, 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 OWL as an IC language, 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. 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.

2 IC Use Cases

There are several common use cases for closed world constraint checking that have been identified in the relational and deductive databases literature [2, Chap. 11]. We prepared a user survey to gather use cases and requirements for ICs from the OWL community. These use cases are similar to what we consider to be the canonical IC use cases and can be summarized as follows:

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. Given the following ICs

$\exists \texttt{hasProducer}.\top \sqsubseteq \texttt{Product}$, $\top \sqsubseteq \forall \texttt{hasProducer}.\texttt{Producer}$

The following role assertion

$\texttt{hasProducer}(\texttt{product1}, \texttt{producer1})$

would violate these ICs since `product1` and `producer1` are not explicitly known to be instances of `Product` and `Producer` respectively. The data would be valid with the addition of the following assertions:

$\texttt{Product}(\texttt{product1})$, $\texttt{Producer}(\texttt{producer1})$.

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 universal restrictions.

Participation constraints Participation constraints require that instances of the constrained class should have a role assertion. Given an IC semantics, the existential restrictions in OWL can be used for this purpose. For instance, in Example 1, α is a participation constraint. With IC semantics, we expect $\mathcal{K}$ to be invalid w.r.t. this constraint since the producer of p is not known. $\mathcal{K}$ would be valid only when additional axioms in the following form are added:

$\texttt{hasProducer}(\texttt{p}, \texttt{producer})$, $\texttt{Producer}(\texttt{producer})$.

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

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. For instance, in Example 2, α is an uniqueness constraint. With IC semantics, $\mathcal{K}$ is invalid w.r.t. this constraint since p has two producers $m1$ and $m2$ which are not known to be same. $\mathcal{K}$ would be valid after adding the assertion $m1 = m2$.

3 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 [3, 4]. For example, according to the proposal in [3], the constraint axiom α in Example 1 is expressed with rules as follows:

$\bot \leftarrow DL[\texttt{Product}](x), \textbf{not}\ P(x, y)$

$P(x, y) \leftarrow DL[\texttt{hasProducer}](x, y), DL[\texttt{Producer}](y)$

where atoms with prefix DL are *DL atoms* which are evaluated as queries to the OWL KB, **not** is the *Negation As Failure (NAF)* operator [3], and $\bot$ is a special predicate representing the empty rule head. The addition of constraints (rules) to a DL KB constitutes a hybird KB, and the detection of a constraint violation is reduced to checking if the special predicate $\bot$ is entailed by the hybrid KB. 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 [5] where EQL-Lite allows one to pose epistemic FOL queries that contain the $\mathcal{K}$ operator used against standard FOL KBs. Since every OWL axiom can be represented as an FOL formula we can translate the constraint axiom in Example 1 to the following EQL-Lite query:

$\mathcal{K}\texttt{Product}(x) \rightarrow \exists y.(\mathcal{K}\texttt{hasProducer}(x, y) \wedge \mathcal{K}\texttt{Producer}(y))$

where the answers of this query return the individuals in the KB that satisfy the constraint, and the answers of the negated query will return the individuals that violate the IC. Although the data complexity of answering domain independent 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 based on the epistemic extension of DLs [6, 7] where modal operators $\mathcal{K}$ and $\mathcal{A}$ can be used in concept and role expressions of the given DL KB. Intuitively, $\mathcal{K}C$ represents the set of individuals that are known to be instances of C and $\mathcal{K}R$ represents the pair of individuals that are known to be related with the role R. Operator $\mathcal{A}$ is interpreted in terms of autoepistemic assumptions. Then the ICs are represented as epistemic DL axioms, and the satisfaction of ICs is defined as the entailment of the epistemic IC axioms by the

[3] NAF is widely used in logic programming systems. With NAF, axioms that cannot be proven to be true are assumed to be false

standard DL KB. For example, the constraint α in Example 1 can be translated into the following epistemic DL axiom:

$\mathcal{K}\texttt{Product} \sqsubseteq \exists \mathcal{K}\texttt{hasProducer}.\mathcal{K}\texttt{Producer}$.

One important feature of [6, 7] is that all interpretation domains are same, and an individual name always refers to the same object in every interpretation. Due to this feature, strict UNA is enforced. That is, two different names always denote different resources. However, this 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. [8] based on a minimal Herbrand model semantics of OWL: here, a constraint axiom is satisfied 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 and its unknown producer, and the constraint α that every product has a known producer:*

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

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

Since p has a producer in every minimal Herbrand model of $\mathcal{K}$, α 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, but still doesn't capture the intuition behind the constraint;

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

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

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

The intuition behind constraint α 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, α is still satisfied by the semantics of [8] 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 also interact in unexpected ways.

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

$\mathcal{K} = \{\texttt{Product} \sqsubseteq \texttt{Category1} \sqcup \texttt{Category2}, \texttt{Product}(p)\}$
$\alpha : \texttt{Category1} \sqsubseteq \exists \texttt{categoryType}.\top$

Since we do not know for sure that p belongs to `Category1`, it is reasonable to assume that the constraint α will not apply to p and α will not be violated. However, with [8] semantics, α is violated because there is a minimal model where p belongs to `Category1` but it does not have a `categoryType` value.

In this paper, we present a new IC semantics for OWL that overcomes the above issues and enables efficient IC validation for OWL.

4 Preliminaries

4.1 Description Logics $\mathcal{SROIQ}$

In this section, we give a brief description about the syntax and semantics of the Description Logic $\mathcal{SROIQ}$ [9], which is the logical underpinning of OWL 2 [10]. More details can be found in [9].

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 Δ is the *domain*, and $\cdot^{\mathcal{I}}$ is the *interpretation function* which maps $A \in N_C$ to a subset of Δ, $R \in N_R$ to a subset of $\Delta \times \Delta$, $a \in N_I$ to an element of Δ. The interpretation can be extended to inverse roles and complex concepts as follows:

$$(R^-)^{\mathcal{I}} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}, (\neg C)^{\mathcal{I}} = \Delta \setminus C^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$$
$$(\geq nR.C)^{\mathcal{I}} = \{x \mid \#\{y.\langle x, y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}$$
$$(\exists R.\text{Self})^{\mathcal{I}} = \{x \mid \langle x, x\rangle \in R^{\mathcal{I}}\}, \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}.$$

where # denotes the cardinality of a set.

A $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ is a collection of $\mathcal{SROIQ}$ axioms, including TBox, RBox, and ABox axioms. A $\mathcal{SROIQ}$-interpretation $\mathcal{I}$ satisfies an axiom α, denoted $\mathcal{I} \models \alpha$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ ($R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$, $R_1^{\mathcal{I}} \circ \ldots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$, $\forall x \in \Delta : \langle x, x\rangle \in R^{\mathcal{I}}$, $\forall x \in \Delta : \langle x, x\rangle \notin R^{\mathcal{I}}$, $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} = \emptyset$ resp.) holds when α=$C \sqsubseteq D$ ($R_1 \sqsubseteq R_2$, $R_1 \ldots R_n \sqsubseteq R$, $\texttt{Ref}(R)$, $\texttt{Irr}(R)$, $\texttt{Dis}(R_1, R_2)$ resp.). Note that, there are also four kinds of ABox axioms ($C(a)$, $R(a, b)$, $a = b$, $a \neq b$). 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.). $\mathcal{I}$ 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* α, written as $\mathcal{K} \models \alpha$, if $\mathcal{I} \models \alpha$ for all models $\mathcal{I} \in Mod(\mathcal{K})$.

4.2 Distinguished Conjunctive Queries (DCQs)

We now describe the syntax and semantics of *distinguished conjunctive queries* (DCQs). 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 the 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 DCQs are given in terms of interpretations defined in Section 4.1. 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 σ 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 σ, 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.

5 IC Semantics for OWL

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 [11]. 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 about the epistemic nature of ICs and believe this is the most appropriate semantics for ICs. In the following section, we describe an alternative IC semantics for OWL axioms, which is similar to how the semantics of epistemic DL $\mathcal{ALCK}$ [6] and MKNF DL $\mathcal{ALCK}_{\mathcal{NF}}$ [7] are defined. Then, in Section 5.2, we discuss how the IC semantics addresses the issues explained in Section 1 and Section 3, and enables OWL to be an IC language.

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

5.1 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 Δ, roles to a subset of $\Delta \times \Delta$ and individuals to an element of Δ 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 is extended to inverse roles and complex concepts as follows:

$$(R^-)^{\mathcal{I},\mathcal{U}} = \{\langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \mid \langle y^{\mathcal{I}}, x^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}}\},$$
$$(C \sqcap D)^{\mathcal{I},\mathcal{U}} = C^{\mathcal{I},\mathcal{U}} \cap D^{\mathcal{I},\mathcal{U}}, \quad (\neg C)^{\mathcal{I},\mathcal{U}} = N_I \setminus C^{\mathcal{I},\mathcal{U}},$$
$$(\geq nR.C)^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \#\{y^{\mathcal{I}} \mid \langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}}, y^{\mathcal{I}} \in C^{\mathcal{I},\mathcal{U}}\} \geq n\},$$
$$(\exists R.\text{Self})^{\mathcal{I},\mathcal{U}} = \{x^{\mathcal{I}} \mid x \in N_I \text{ s.t. } \langle x^{\mathcal{I}}, x^{\mathcal{I}}\rangle \in R^{\mathcal{I},\mathcal{U}}\}, \{a\}^{\mathcal{I},\mathcal{U}} = \{a^{\mathcal{I}}\}.$$

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 $\mathcal{ALCK}_{\mathcal{NF}}$ [6, 7], 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 [6, 7] is limited to $\mathcal{ALC}$. Second, in $\mathcal{ALCK}$ and $\mathcal{ALCK}_{\mathcal{NF}}$ [6, 7], 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.

An IC-interpretation $\mathcal{I}$,$\mathcal{U}$ satisfies an axiom α, denoted as $\mathcal{I}$,$\mathcal{U} \models \alpha$, if $C^{\mathcal{I},\mathcal{U}} \subseteq D^{\mathcal{I},\mathcal{U}}$ ($R_1^{\mathcal{I},\mathcal{U}} \subseteq R_2^{\mathcal{I},\mathcal{U}}$, $R_1 \sqsubseteq R_2$, $R_1^{\mathcal{I},\mathcal{U}} \subseteq R_2^{\mathcal{I},\mathcal{U}}$, $\forall x \in N_I : \langle x^{\mathcal{I},\mathcal{U}}, x^{\mathcal{I},\mathcal{U}} \rangle \in R^{\mathcal{I},\mathcal{U}}$, $\forall x \in N_I : \langle x^{\mathcal{I},\mathcal{U}}, x^{\mathcal{I},\mathcal{U}} \rangle \notin R^{\mathcal{I},\mathcal{U}}$, $R_1^{\mathcal{I},\mathcal{U}} \cap R_2^{\mathcal{I},\mathcal{U}} = \emptyset$ resp.) holds when α=$C \sqsubseteq D$ ($R_1 \sqsubseteq R_2$, $R_1 \ldots R_n \sqsubseteq R$, $\texttt{Ref}(R)$, $\texttt{Irr}(R)$, $\texttt{Dis}(R_1, R_2)$ resp.).

Given a $\mathcal{SROIQ}$ KB $\mathcal{K}$ and a $\mathcal{SROIQ}$ constraint α, the IC-satisfaction of α by $\mathcal{K}$, i.e., $\mathcal{K} \models_{IC} \alpha$, is defined as:

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

We define an *extended KB* as a pair $\langle \mathcal{K}, \mathcal{C} \rangle$ where $\mathcal{K}$ is a $\mathcal{SROIQ}$ KB as before and $\mathcal{C}$ is a set of $\mathcal{SROIQ}$ axioms interpreted with IC 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.

5.2 Discussion

It is easy to verify that the IC semantics provides expected results for the examples presented in Section 1 and Section 3. For instance, we get an IC violation in Example 1 since the IC interpretation of `Product` contains p but the IC interpretation of ($\exists$`hasProducer`.`Producer`) is empty.

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 α:*

$$\mathcal{K} = \{C(c), R(c, d_1), R(c, d_2), D(d_1), D(d_2)\}, \quad \alpha : C \sqsubseteq \geq 2R.D$$

With the weak UNA, 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 α 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 α:*

$$\mathcal{K} = \{C(a), (C_1 \sqcup C_2)(a)\}, \quad \alpha : C \sqsubseteq C_1 \sqcup C_2$$

Constraint α 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, α 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 α' as follows:

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

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.

6 IC Validation

We have defined in Section 5.1 that, the extended KB $\langle \mathcal{K}, \mathcal{C} \rangle$ is valid if 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 NAF operator **not** . We start by giving the formal semantics for **not** in DCQs, then describe the translation rules from IC axioms to DCQ$^{\mathbf{not}}$ and finally provide a theorem showing that IC validation can be reduced to answering DCQ$^{\mathbf{not}}$ under certain conditions.

6.1 DCQ$^{\mathbf{not}}$

In Section 4.2 , we introduced standard DCQs. However, the expressivity of standard DCQs is not enough to capture the closed world nature of IC semantics. For this reason, we add the **not** operator to DCQs to get DCQ$^{\mathbf{not}}$ queries. The syntax of DCQ$^{\mathbf{not}}$ is defined as follows:

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

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

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

And we use the abbreviation $Q_1 \vee Q_2$ for **not** (**not** $Q_1 \wedge$ **not** Q_2). We can see

$$\mathcal{K} \models^{\sigma} Q_1 \vee Q_2 \qquad \text{iff} \qquad \mathcal{K} \models^{\sigma} Q_1 \text{ or } \mathcal{K} \models^{\sigma} Q_2$$

6.2 Translation Rules: from ICs to DCQ$^{\mathbf{not}}$

We now present the translation rules from IC axioms to DCQ$^{\mathbf{not}}$ queries. The translation rules are similar in the spirit to the Lloyd-Topor transformation [12]

but instead of rules we generate DCQ$^{\mathbf{not}}$ queries. 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 to 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$^{\mathbf{not}}$ query as the result:

$$\mathcal{T}_c(C_a, x) := C_a(x)$$
$$\mathcal{T}_c(\neg C, x) := \mathbf{not}\, \mathcal{T}_c(C, x)$$
$$\mathcal{T}_c(C_1 \sqcap C_2, x) := \mathcal{T}_c(C_1, x) \wedge \mathcal{T}_c(C_2, x)$$
$$\mathcal{T}_c(\geq nR.C, x) := \bigwedge_{1 \leq i \leq n} (R(x, y_i) \wedge \mathcal{T}_c(C, y_i)) \bigwedge_{1 \leq i < j \leq n} \mathbf{not}\, (y_i = y_j)$$
$$\mathcal{T}_c(\exists R.\mathrm{Self}, x) := R(x, x)$$
$$\mathcal{T}_c(\{a\}, x) := (x = a)$$

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

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

$$\mathcal{T}(C_1 \sqsubseteq C_2) := \mathcal{T}_c(C_1, x) \wedge \mathbf{not}\, \mathcal{T}_c(C_2, x)$$
$$\mathcal{T}(R_1 \sqsubseteq R_2) := R_1(x, y) \wedge \mathbf{not}\, R_2(x, y)$$
$$\mathcal{T}(R_1 \ldots R_n \sqsubseteq R) := R_1(x, y_1) \wedge \ldots R_n(y_{n-1}, y_n) \wedge \mathbf{not}\, R(x, y_n)$$
$$\mathcal{T}(\mathtt{Ref}(R)) := \mathbf{not}\, R(x, x)$$
$$\mathcal{T}(\mathtt{Irr}(R)) := R(x, x)$$
$$\mathcal{T}(\mathtt{Dis}(R_1, R_2)) := R_1(x, y) \wedge R_2(x, y)$$

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

6.3 Reducing IC Validation to Answering DCQ$^{\mathbf{not}}$

In Theorem 1, we show that IC validation via query answering is sound and complete when the expressivity of the extended KB is either $\langle \mathcal{SRI}, \mathcal{SROIQ} \rangle$ or $\langle \mathcal{SROIQ}, \mathcal{SROI} \rangle$. Note that, when the expressivity is $\langle \mathcal{SROIQ}, \mathcal{SROIQ} \rangle$, we can not reduce IC validation to query answering in a straightforward way due to the interaction between the disjunctive (in)equality axioms in $\mathcal{K}$ and the cardinality constraints in $\mathcal{C}$. We limit this interaction by either excluding nominals and cardinality restrictions in $\mathcal{K}$ thus prohibiting disjunctive (in)equality to appear in $\mathcal{K}$, or by prohibiting cardinality restrictions in $\mathcal{C}$. Due to space limitations we only present the main theorem here. The complete proofs are presented in the technical report [13].

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 have that $\mathcal{K} \models_{IC} \alpha$ iff $\mathcal{K} \not\models \mathcal{T}(\alpha)$ where $\alpha \in \mathcal{C}$.*

7 Implementation

The emerging best practice query language for OWL ontologies is SPARQL [14] which is known to have the same expressive power as nonrecursive Datalog programs [15] and can express DCQ$^{\mathbf{not}}$ queries. Therefore, based on the results from Section 6.3, we can reduce IC validation to SPARQL query answering if the KB is $\mathcal{SRI}$ or the ICs do not contain cardinality restrictions.

We have built a prototype IC validator[5] by extending the OWL 2 DL reasoner Pellet[6]. The prototype reads ICs expressed as OWL axioms and translates each IC first to a DCQ$^{\mathbf{not}}$ query and then to a SPARQL query. The resulting query 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 proof-of-concept prototype to validate ICs with several large ontologies such as the LUBM dataset.[7] 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.

8 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$^{\mathbf{not}}$ queries, 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 $\mathcal{SROIQ}$.

[5] http://clarkparsia.com/pellet/oicv-0.1.2.zip
[6] http://clarkparsia.com/pellet
[7] http://swat.cse.lehigh.edu/projects/lubm/

References

1. Smith, M.K., Welty, C., McGuiness, D.: OWL web ontology language guide (2004)
2. Colomb, R.M.: Deductive Databases and Their Applications. CRC Press (1998)
3. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the semantic web. AI **172**(12-13) (2008) 1495–1539
4. Motik, B.: A faithful integration of description logics with logic programming. In: IJCAI2007. (2007) 477–482
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: EQL-Lite: Effective first-order query processing in description logics. In: IJCAI2007. (2007) 274–279
6. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A.: An epistemic operator for description logics. AI **100**(1–2) (1998) 225–274
7. Donini, F.M., Nardi, D., Rosati, R.: Description logics of minimal knowledge and negation as failure. ACM Transactions on Computational Logic **3** (2002) 177–225
8. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. In: WWW2007. (2007) 807–816
9. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: KR2006. (2006) 57–67
10. Motik, B., Patel-Schneider, P.F., Grau, B.C.: OWL 2 web ontology language direct semantics (2009)
11. Reiter, R.: On integrity constraints. In: TARK1988. (1988) 97–111
12. Lloyd, J.W.: Foundations of logic programming. (1987)
13. Tao, J., Sirin, E., Bao, J., McGuinness, D.: Integrity constraints in OWL. Technical report, Rensselaer Polytechnic Institute, Troy, NY, USA (2010)
14. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2008)
15. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: ISWC2008. (2008) 114–129