Object-Relational Queries over $\mathcal{CFDI}_{nc}^{\forall-}$ Knowledge Bases: OBDA for the SQL-Literate

Jason St. Jacques, David Toman, and Grant Weddell
Cheriton School of Computer Science, University of Waterloo, Canada
jw.stjacques@gmail.com, {david,gweddell}@cs.uwaterloo.ca

Abstract

We consider how SQL-like query languages over object-relational schemata can be preserved in the setting of *ontology based data access* (OBDA), thus leveraging wide familiarity with relational technology. This is enabled by the adoption of the logic $\mathcal{CFDI}_{nc}^{\forall-}$, a member of the $\mathcal{CFD}$ family of *description logics* (DLs). Of particular note is that this logic can fully simulate DL-Lite$_{core}^{\mathcal{F}}$, a member of the DL-Lite family commonly used in the OBDA setting. Our main results present efficient algorithms that allow computation of certain answers with respect to $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases, facilitating direct access to a pre-existing row-based relational encoding of the data without any need for mappings to triple-based representations.

1 Introduction

Ontology based data access (OBDA) is concerned with computing query answers over (possibly incomplete) data sources for which *background knowledge* about the data, commonly captured in an ontology, is available. The background knowledge provides additional query answers that may not be explicit in the data itself. To address scalability issues relating to the volume of data, many current approaches to OBDA focus on conjunctive queries (CQ) and ontologies based on DL dialects for which CQ answering is in AC^0/PTIME with respect to data complexity. Moreover, to leverage advances in query processing in relational systems, approaches in which query answering can be reduced to SQL query evaluation over a relational encoding of the data are commonly sought. The two front-runners in this area are (i) the *perfect rewriting*-based approaches in which the given CQ is rewritten with the help of the ontological knowledge (typically formulated in one of the DL-Lite family of logics) in such a way that the resulting query can be executed over the original data sources to obtain desired answers [Calvanese *et al.*, 2007], and (ii) the *combined* approaches in which the data is completed using ontological knowledge (formulated in DL-Lite or $\mathcal{EL}$ logics) in such a way that the original query (modulo filtering that only depends on role hierarchies in the ontology) can then be executed over the data completion [Kontchakov *et al.*, 2010; 2011; Lutz *et al.*, 2013; 2009]. The closest to our approach is the approach for Horn-$\mathcal{SHIQ}$ with rules [Eiter *et al.*, 2012]; that approach deals with a logic incomparable with $\mathcal{CFDI}_{nc}^{\forall-}$. Moreover, our approach handles keys and functional dependencies that are essential in database applications.

Recently, Toman and Weddell proposed $\mathcal{CFDI}_{nc}^{\forall-}$ [Toman and Weddell, 2014], a dialect of the $\mathcal{CFD}$ family [Khizder *et al.*, 2000; Toman and Weddell, 2009; 2013] that has PTIME complexity for many of the fundamental reasoning tasks, and can fully simulate DL-Lite$_{core}^{\mathcal{F}}$ [Toman and Weddell, 2015].

In this paper, we show that CQ answering over $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge bases can be reduced to evaluating SQL queries over (a completion of) the data stored in a relational system. Our technique is based on a *combination* of query rewriting and data completion. Indeed, it is worth noting that, for CQs over $\mathcal{CFDI}_{nc}^{\forall-}$ KBs, OBDA *cannot* be accomplished by either using (perfect) query rewriting alone, due to PTIME-completeness of CQ answering, or by exclusive use of the combined approach, due to the need to realize exponentially many prototypical anonymous witnesses to represent types induced by value restrictions. Our main technical contributions, in the order presented, are as follows:

- We exhibit an ABox completion procedure for a given *logic* knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with PTIME data complexity; the completion also serves as a basis for KB consistency checking;
- We define a query rewriting that produces a union of conjunctive queries Q' from a given conjunctive query Q and $\mathcal{T}$, and show that evaluating Q' as a SQL query over the above ABox completion, viewed as a relational database, computes the certain answers of Q over $\mathcal{K}$.
- Finally we show how a standard relational database schema can be naturally captured as a (fragment of a) $\mathcal{CFDI}_{nc}^{\forall-}$ TBox, eliminating the need for additional *mappings* between data sources and virtual ABoxes that are typically utilized at this point, e.g., by [Calvanese *et al.*, 2015]. We then show how such a rewritten query can be executed over an underlying relational representation *without* the need for object (id) invention.

Moreover, we show that the potential exponential blowup of the query rewriting cannot be avoided in general, since the combined complexity for CQ answering in $\mathcal{CFDI}_{nc}$ is PSPACE-complete (unless NP=PSPACE). However, the exponential blowup originating from concept hierarchies is

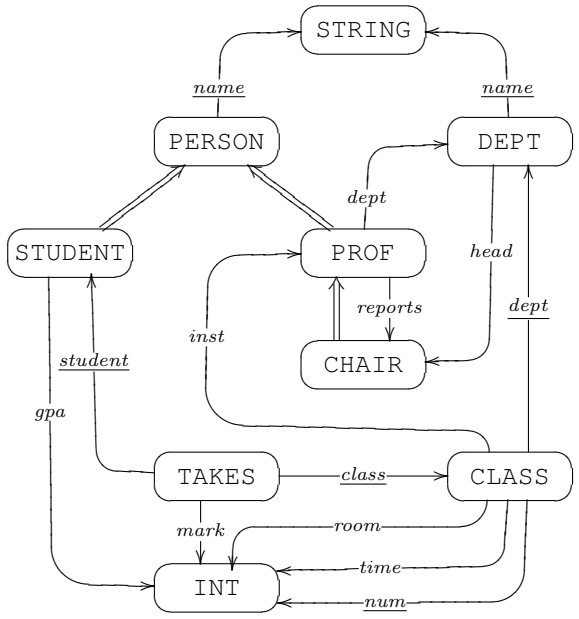

Figure 1: UNIV Schema.

eliminated/minimized thanks to the data completion.

Example 1 In the rest of the paper we use an object-relational UNIV schema depicted in Figure 1 in which single arrows denote named features, double arrows inheritance between classes, and where primary keys are underlined. We can then query instances of this schema using familiar SQL/OQL-style syntax for queries, for example:

```
select distinct s.name as n
from STUDENT s, TAKES t, CLASS c
where s = t.student and
      c.time = t.class.time and
      c.inst.dept.name = :p1 and c.num = :p2
```

The remainder of the paper is organized as follows: preliminaries are given next in which we introduce $\mathcal{CFDI}_{nc}^{\forall-}$ and conjunctive queries. In Section 3, we present the theoretical basis of our approach, and in Section 4 discuss practical matters connected with the use of a relational system. We then outline some preliminary experimental results and follow with a summary, including some brief comments on directions for future work.

2 Preliminaries

All members of the $\mathcal{CFD}$ family of DLs are fragments of FOL with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *features*. Formally, the logics are defined with respect to the sets F, PC and IN of (names of) features, primitive concepts, and individuals, respectively. A *path function* Pf is a word in F^* with the usual convention that the empty word is denoted by id and concatenation by ".". *Concept descriptions* C and D are defined by the grammars on the left-hand-side of Figure 2. A concept "$\mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$" is called a *path functional dependency* (PFD) and must conform to the following forms:

SYNTAX	SEMANTICS: "$(\cdot)^{\mathcal{I}}$"
$\mathrm{C} ::= \mathrm{A}$	$\mathrm{A}^{\mathcal{I}} \subseteq \triangle$
$\mid \forall \mathsf{Pf}.\mathrm{C}$	$\{x \mid \mathsf{Pf}^{\mathcal{I}}(x) \in \mathrm{C}^{\mathcal{I}}\}$
$\mid \exists f^{-1}$	$\{x \mid \exists y \in \triangle : f^{\mathcal{I}}(y) = x\}$
$\mathrm{D} ::= \mathrm{C}$	$\mathrm{C}^{\mathcal{I}} \subseteq \triangle$
$\mid \neg \mathrm{C}$	$\triangle \setminus \mathrm{C}^{\mathcal{I}}$
$\mid \forall \mathsf{Pf}.\mathrm{D}$	$\{x \mid \mathsf{Pf}^{\mathcal{I}}(x) \in \mathrm{D}^{\mathcal{I}}\}$
$\mid \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$	$\{x \mid \forall y \in \mathrm{C}^{\mathcal{I}} : (\bigwedge_{i=1}^{k} \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y)) \Rightarrow \mathsf{Pf}^{\mathcal{I}}(x) = \mathsf{Pf}^{\mathcal{I}}(y)\}$

Figure 2: $\mathcal{CFDI}_{nc}$ Concepts.

1. $\mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.\,\mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$ or
2. $\mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.f, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.g$

Semantics is defined in the standard way with respect to an interpretation $\mathcal{I} = (\triangle, (\cdot)^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretation of primitive concepts A to be subsets of $\triangle$, features $f \in \mathsf{F}$ to be total functions on $\triangle \rightarrow \triangle$, and individuals a to be elements of $\triangle$. The interpretation function is extended to path expressions by interpreting id as the identity function $\lambda x.x$, concatenation as function composition, and to derived concept descriptions C or D as defined in Figure 2.

An interpretation $\mathcal{I}$ satisfies an *inclusion dependency* $\mathrm{C} \sqsubseteq \mathrm{D}$ if $\mathrm{C}^{\mathcal{I}} \subseteq \mathrm{D}^{\mathcal{I}}$, a *concept assertion* $\mathrm{A}(a)$ if $a^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$, and a *path function assertion* $\mathsf{Pf}_1(a) = \mathsf{Pf}_2(b)$ if $\mathsf{Pf}_1^{\mathcal{I}}(a^{\mathcal{I}}) = \mathsf{Pf}_2^{\mathcal{I}}(b^{\mathcal{I}})$. $\mathcal{I}$ satisfies a knowledge base $\mathcal{K}$ if it satisfies each inclusion dependency and assertion in $\mathcal{K}$.

It is easy to see that for every $\mathcal{CFDI}_{nc}$ KB $(\mathcal{T}, \mathcal{A})$, there is a conservative extension $(\mathcal{T}', \mathcal{A}')$ in which subsumptions in $\mathcal{T}'$ adhere to the following forms:

$$\mathrm{A} \sqsubseteq \mathrm{B},\ \ \mathrm{A} \sqsubseteq \forall f.\mathrm{B},\ \ \forall f.\mathrm{A} \sqsubseteq \mathrm{B},\ \ \mathrm{A} \sqsubseteq \exists f^{-1},\ \text{or}$$
$$\mathrm{A} \sqsubseteq \mathrm{A}' : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf},$$

where A and A′ are primitive concepts, B is a primitive concept or a negation of a primitive concept, and $f \in \mathsf{F}$ a feature. Similarly, the ABox $\mathcal{A}'$ contains only assertions of the form "$\mathrm{A}(a)$", "$a.f = b$", and "$a = b$".

For detailed description of $\mathcal{CFDI}_{nc}$ and its variants see [Toman and Weddell, 2014]. The following proposition summarizes computational properties of $\mathcal{CFDI}_{nc}^{\forall-}$ pertinent to the development in this paper:

Proposition 2 ($\mathcal{CFDI}_{nc}^{\forall-}$ Properties) Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base, $A_1, \ldots, A_k$ primitive concepts, C a concept description not containing the PFD constructor, and D a concept description. Then *logical implication*, $\mathcal{T} \models C \sqsubseteq D$, and $\mathcal{K}$ *consistency* can be decided in PTIME. Moreover, the question of whether or not a conjunction of concepts C_i $(i \leq n)$ interprets as non-empty in every model of $\mathcal{K} = (\mathcal{T}, \{A_1(a), \ldots, A_k(a)\})$ (i.e., when $A_1, \ldots, A_k$ *forces* $C_1, \ldots, C_n$) is PTIME-complete in $|\mathcal{K}|$ and PSPACE-complete in n.

Example 3 Sample constraints for the UNIV schema de-

picted in Figure 1 can be captured in $\mathcal{CFDI}_{nc}^{\forall-}$ as follows:

1. PERSON $\sqsubseteq \neg$DEPT,
2. PERSON $\sqsubseteq \forall name.$STRING,
3. PERSON $\sqsubseteq$ PERSON : $name \to id$,
4. PERSON $\sqsubseteq$ DEPT : $name \to id$,
5. PROF $\sqsubseteq$ PERSON,
6. $\forall reports.$CHAIR $\sqsubseteq$ PROF,
7. $\exists head^{-1} \sqsubseteq$ CHAIR,
8. CLASS $\sqsubseteq$ CLASS : $dept, num \to id$, etc.

2.1 Conjunctive Queries and Certain Answers

Conjunctive queries are, as usual, formed from atomic queries (or *atoms*) of the form "$C(x)$" and "$x.\mathsf{Pf}_1 = y.\mathsf{Pf}_2$", where x and y are variables, using conjunction and existential quantification. To simplify notation, we conflate conjunctive queries with the set of its constituent atoms and a set of *answer variables*:

Definition 4 (Conjunctive Query) Let φ be a set of atoms $C(x_i)$ and $x_{i_1}.\mathsf{Pf}_1 = x_{i_2}.\mathsf{Pf}_2$, where C is a concept description (defined in Figure 2), Pf_i are path functions, and $\bar{x}$ a tuple of variables. We call the expression $\{\bar{x} \mid \varphi\}$ a *conjunctive query* (CQ).

A conjunctive query "$\{\bar{x} \mid \varphi\}$" is therefore a notational variant of the formula "$\exists \bar{y}. \bigwedge_{\psi \in \varphi} \psi$" in which $\bar{y}$ contains all variables appearing in φ but not in $\bar{x}$. The usual definition of certain answers is given by the following:

Definition 5 (Certain Answer) Let $\mathcal{K}$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base and $Q = \{\bar{x} \mid \varphi\}$ a CQ. A *certain answer* to Q over $\mathcal{K}$ is a substitution of constant symbols $\bar{a}$, $[\bar{x} \mapsto \bar{a}]$, such that $\mathcal{K} \models Q[\bar{x} \mapsto \bar{a}]$.

As is the case with TBoxes and ABoxes, a CQ can be represented in a *normal form*, a form in which all atoms in the CQ are of the form "$A(x)$" or "$x.f = y$", where A is a primitive concept and f a feature. This can be easily achieved by introducing additional non-answer (existentially quantified) variables.

Example 6 The OQL query from Example 1 above is now formally captured as the following CQ in normal form:

$$\{\langle n, p1, p2\rangle \mid \mathrm{STUDENT}(s), \mathrm{TAKES}(t), \mathrm{CLASS}(c), \\ n = s.name, p2 = c.num, p1 = d'.name, \\ c' = t.class, p' = c.inst, d' = p'.dept, \\ s = t.student, t' = c.time, t' = c'.time\}.$$

For the remainder of the paper, we assume CQs are always *connected*. (Evaluating disconnected CQs is easily achieved by considering each component separately.)

3 Query Answering (OBDA)

We begin by introducing OBDA over *abstract* ABoxes. From the introduction: we need a *first step* to obtain an ABox completion $\mathcal{A}_\mathcal{T}$ depending only on $\mathcal{A}$ and $\mathcal{T}$ that is polynomial in both $|\mathcal{A}|$ and $|\mathcal{T}|$, and a *second step* to obtain a query rewriting $Q_\mathcal{T}$ depending only on Q and $\mathcal{T}$ that is polynomial in $|\mathcal{T}|$, such that $\bar{a}$ is a certain answer to Q in $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if and only if $\bar{a}$ is an answer to $Q_\mathcal{T}$ when evaluated over $\mathcal{A}_\mathcal{T}$.

3.1 Inverse-Induced PFDs

Allowing *inverse features* affects how PFDs interact with an ABox. In particular, PFDs in which all path functions have a common prefix may apply to (pairs of) anonymous individuals mandated by the existence of anonymous inverse features for existing ABox individuals. Consider the following example:

Example 7 For an ABox $\mathcal{A} = \{A(a), A(b), a.g = b.g\}$ and a TBox $\mathcal{T} = \{A \sqsubseteq \exists f^{-1}, \forall f.A \sqsubseteq B, B \sqsubseteq B : f.g \to id\}$, it is easy to see that both a and b must have an f-predecessor to which the PFD in $\mathcal{T}$ applies and, in combination with functionality of features, yields $a = b$.

Observe with this example, however, that the TBox logically implies another PFD, in particular "$A \sqsubseteq A : g \to id$", and that this PFD *applies* to objects *explicitly* present in $\mathcal{A}$. In general, to enforce PFDs on an ABox *while avoiding any need to explicitly create anonymous predecessor objects*, we add additional logically implied PFDs to a given TBox as follows:

Definition 8 (PFD Enrichment for Inverses) Let $\mathcal{T}$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ TBox such that $\mathrm{A} \sqsubseteq \mathrm{B} : f.\mathsf{Pf}_1, \ldots, f.\mathsf{Pf}_k \to f.\mathsf{Pf} \in \mathcal{T}$ ($\mathrm{A} \sqsubseteq \mathrm{B} : f.\mathsf{Pf}_1, \ldots, f.\mathsf{Pf}_k \to id \in \mathcal{T}$) where $\mathsf{Pf}_i \neq id$ for all $1 \leq i \leq k$. Then we require that $\mathrm{A} \sqsubseteq \forall f.\mathrm{A}', \mathrm{B} \sqsubseteq \forall f.\mathrm{B}'$, and $\mathrm{A}' \sqsubseteq \mathrm{B}' : \mathsf{Pf}_1, \ldots, f.\mathsf{Pf}_k \to \mathsf{Pf}$ ($\mathrm{A}' \sqsubseteq \mathrm{B}' : \mathsf{Pf}_1, \ldots, f.\mathsf{Pf}_k \to id$), where A' and B' are fresh primitive concepts, are also in $\mathcal{T}$.

It is easy to see that the value restrictions and PFDs added to $\mathcal{T}$ logically follow from the original PFD and the necessary existence of f-inverses for A' and B' that originate in A and B, respectively. Thus, we assume that TBoxes satisfy this condition for the remainder of the paper.

3.2 Abstract ABox Completion

The first step of query answering, ABox completion, is defined by the rules in Figure 3. In particular, the rules extend a given ABox with all implied concept memberships and feature agreements. (Note that this step *cannot be accomplished* by a FO query since it requires *path exploration* and is therefore hard for NLOGSPACE.)

Definition 9 Let $(\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFDI}_{nc}^{\forall-}$ knowledge base. We define an ABox $\mathsf{completion}_\mathcal{T}(\mathcal{A})$ to be the least ABox $\mathcal{A}'$ such that $\mathcal{A} \subseteq \mathcal{A}'$ and $\mathcal{A}'$ is closed under the rules in Figure 3.

Observe that individuals can only be declared to be members of primitive concepts since $\mathcal{A}$ is in normal form. Also, if UNA were to be assumed (or assumed for elements of specific primitive concepts, such as INT and STRING), the equalities in rules (a) and (c) lead to KB inconsistency (without any need for additional "firing of rules"). It is also easy to see that completion terminates since it can add at most $|\mathcal{T}||A|^2$ new objects, one for every pair of existing objects and a feature name.

3.3 Query Rewriting

The second step in query answering relies on query reformulation with respect to $\mathcal{T}$. This step is also necessary as $\mathcal{CFDI}_{nc}^{\forall-}$ can force *exponentially many* anonymous objects *with distinct class membership* to exist:

if $a = b \in \mathcal{A}$ then add $b = a$ to $\mathcal{A}$
if $a = b, \varphi \in \mathcal{A}$ then add $\varphi[b/a]$ to $\mathcal{A}$

(a) ABox Equality Interactions

if $\mathrm{A}(a) \in \mathcal{A}$ and $\mathcal{T} \models \mathrm{A} \sqsubseteq \mathrm{B}$ then add $\mathrm{B}(a)$ to $\mathcal{A}$
if $\{\mathrm{A}(a), a.f = b\} \subseteq \mathcal{A}$ and $\mathcal{T} \models \mathrm{A} \sqsubseteq \forall f.\mathrm{B}$ then add $\mathrm{B}(b)$ to $\mathcal{A}$
if $\{\mathrm{A}(a), b.f = a\} \subseteq \mathcal{A}$ and $\mathcal{T} \models \forall f.\mathrm{A} \sqsubseteq \mathrm{B}$ then add $\mathrm{B}(b)$ to $\mathcal{A}$

(b) ABox–TBox Interactions

if $A(a), B(b) \in \mathcal{A}$, $a.\mathsf{Pf}'_i = c_i, b.\mathsf{Pf}'_i = c_i \in \mathcal{A}$ for $0 < i \leq k$, and $\mathrm{A} \sqsubseteq \mathrm{B} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf} \in \mathcal{T}$ then

1. if $a.\mathsf{Pf} = c, b.\mathsf{Pf} = d \in \mathcal{A}$ and $c = d \notin \mathcal{A}$ then add $c = d$ to $\mathcal{A}$; or
2. if Pf is of the form $\mathsf{Pf}''.f$ and $a.\mathsf{Pf}'' = c, b.\mathsf{Pf}'' = d$ and $c = d \notin \mathcal{A}$ then add $c.f = e, d.f = e$ to $\mathcal{A}$;

where Pf'_i is a prefix of Pf_i, c and d are $\mathcal{A}$ individuals, and e is a new individual.

(c) ABox–PFD Interactions

Figure 3: ABox Completion Rules.

Example 10 Consider a $\mathcal{CFDI}_{nc}$ TBox

$$\mathcal{T} = \{\mathrm{B}_i \sqsubseteq \forall \underbrace{f.\cdots.f}_{i^{th}\text{prime}}.\mathrm{B}_i \mid i \leq k\}.$$

Asserting $(\mathrm{B}_0 \sqcap \ldots \sqcap \mathrm{B}_k)(a)$ would require exponentially many anonymous objects belonging to distinct concept combinations to be created as prototypical witnesses when completing the ABox along the lines of the combined approach [Lutz *et al.*, 2009].

To avoid the need for "expensive" ABox completion, our approach treats matches to anonymous individuals by query reformulation along the lines of [Calvanese *et al.*, 2007]:

Definition 11 Let $Q = \{\bar{x} \mid \varphi\}$ be a CQ. We write $\mathsf{Fold}_{\mathcal{T}}(Q)$ to denote the set of CQs (implicitly denoting the union of their results) that is obtained by an exhaustive application of the following on the initial set $\{\{\bar{x} \mid \varphi\}\}$.

For a CQ $\{\bar{y} \mid \psi\} \in \mathsf{Fold}(Q)$, apply rewrite rules:

1. If $\{\mathrm{A}(x), \mathrm{B}(x)\} \subseteq \psi$ and $\mathcal{T} \models \mathrm{A} \sqsubseteq \neg\mathrm{B}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) - \{\{\bar{y} \mid \psi\}\}$.
2. If $\{x.f = y, x.f = z\} \subseteq \psi$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) - \{\{\bar{y} \mid \psi\}\} \cup \{\{\bar{y} \mid \psi\}[z/y]\}$.
3. If $\{x.f = z, y.f = z\} \subseteq \psi$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi\}[x/y]\}$.
4. If $\{\mathrm{A}(x), \mathrm{B}(x)\} \subseteq \psi$ and $\mathcal{T} \models \mathrm{A} \sqsubseteq \mathrm{B}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) - \{\{\bar{y} \mid \psi\}\} \cup \{\{\bar{y} \mid \psi - \{\mathrm{A}(x)\}\}\}$.
5. If $\{x.f = y, \mathrm{A}_1(y), \ldots, \mathrm{A}_k(y)\} \subseteq \psi$ and y does not appear elsewhere in ψ nor in $\bar{y}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi'\}\}$ for all $\psi' = \psi - \{x.f = y, \mathrm{A}_1(y), \ldots, \mathrm{A}_k(y)\} \cup \{\mathrm{B}_{1,i_1}(x), \ldots, \mathrm{B}_{k,i_k}(x)\}$ for which $\mathcal{T} \models \mathrm{B}_{i,i_j} \sqsubseteq \forall f.\mathrm{A}_i$ and B_{i,i_j} is maximal w.r.t. $\sqsubseteq$ for each i.
6. If $\{y.f = x, \mathrm{A}_1(y), \ldots, \mathrm{A}_k(y)\} \subseteq \psi$ and y does not appear elsewhere in ψ nor in $\bar{y}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi'\}\}$, for all $\psi' = \psi - \{y.f = x, \mathrm{A}_1(y), \ldots, \mathrm{A}_k(y)\} \cup \{\mathrm{B}_{1,i_1}(x), \ldots, \mathrm{B}_{k,i_k}(x)\}$ such that $\mathcal{T} \models \forall f.\mathrm{B}_{i,i_j} \sqsubseteq \mathrm{A}_i$, where, for each i, B_{i,i_j} is maximal w.r.t. $\sqsubseteq$, and for which $\mathcal{T} \models \mathrm{B}_{i,i_j} \sqsubseteq \exists f^{-1}$ some $1 \leq i \leq k$.

The key idea underlying this definition is that, to find query answers, it is now sufficient to *match* queries in $\mathsf{Fold}_{\mathcal{T}}(Q)$ explicitly against the (extended) ABox; matches outside the ABox are captured by query reformulation: removed parts of the query are implied by $\mathcal{T}$.

Lemma 12 Let Q be a CQ with at least one answer variable. Then $\bar{a}$ is a certain answer to Q over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if and only if $\bar{a}$ is an answer for some $\{\bar{x} \mid \psi\} \in \mathsf{Fold}_{\mathcal{T}}(Q)$ over $\mathcal{A}_{\mathcal{T}}$.

Proof (sketch): Observing that the extended ABox $\mathcal{A}_{\mathcal{T}}$ is essentially a part of the minimal model of $\mathcal{K}$ (since $\mathcal{K}$ is Horn) and that every element of $\mathsf{Fold}_{\mathcal{T}}(Q)$ implies Q, it is easy to see that whenever (1-6) are satisfied, there is a match of Q in the minimal model and thus $\bar{a}$ is an answer. Conversely, if a match of Q in a minimal model exists yielding $\bar{a}$ as an answer, then part of the match will be realized in the ABox (since at least the answer variables must be bound to ABox individuals) and the reminder of the match must be forest-like. Hence, one of the queries in $\mathsf{Fold}_{\mathcal{T}}(Q)$ matches in the ABox since the remaining conjuncts must be implied by $\mathcal{T}$. □

Example 13 $\mathsf{Fold}_{\mathcal{T}}(Q)$, where Q is the normal form query in Example 6, will contain the following CQ

$$\{\langle n, p1, p2\rangle \mid \mathrm{STUDENT}(s), \mathrm{TAKES}(t), \mathrm{CLASS}(c), \\ n = s.name, p2 = c.num, p1 = d'.name, \\ c = t.class, p' = c.inst, d' = p'.dept, \\ s = t.student\},$$

which is obtained from Q by applying rules (3) and (5).

For CQ without answer variables, we need an additional step that checks if the query (when equivalent to a concept) matches in the tree part of every interpretation of $\mathcal{K}$. We therefore extend the query rewriting in Definition 11 as follows:

7. If $\bar{x} = \langle\rangle$ in $Q = \{\bar{x} \mid \varphi\}$ and φ is equivalent to a conjunction of concepts $\mathrm{C}_1, \ldots, \mathrm{C}_n$. Then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\langle\rangle \mid \mathrm{A}_1(x) \wedge \ldots \wedge \mathrm{A}_k(x)\}$ for every combination of primitive concepts $\mathrm{A}_1, \ldots, \mathrm{A}_k$ that force $\mathrm{C}_1, \ldots, \mathrm{C}_n$ in $\mathcal{T}$.

This construction accounts for matches in the anonymous part of the minimal model of $\mathcal{K}$, and yields the following Lemma:

Lemma 14 Let Q be a CQ without answer variables. Then $\mathcal{K} \models Q$ if and only if at least one $\{\langle\rangle \mid \psi\} \in \mathsf{Fold}_{\mathcal{T}}(Q)$ evaluates to true over $\mathcal{A}_{\mathcal{T}}$.

3.4 Anonymous Object Invention

To handle equalities generated by PFDs, ABox completion, in particular completion rule (c.2) in Figure 3, requires inventing new ABox individuals (denoted e in the figure). These individuals then allow agreement in queries to be resolved within the extended ABox. However, in many situations, individual invention (which, in turn leads to primary key invention as we will see in Section 4) may not be possible in many practical applications (indeed, Section 4 will rely on this).

Here, we explore an alternative based on query reformulation. Unfortunately, it is easy to see for the PFDs of the second form (those that lead to individual invention), that this is not possible. In particular, consider the following case:

Example 15 Assume $\mathcal{T} = \{\mathrm{A} \sqsubseteq \forall f.\mathrm{A}, \mathrm{A} \sqsubseteq \mathrm{A} : f.h, g \rightarrow h\}$ and consider the CQ $\{\langle x, y\rangle \mid \mathrm{A}(x), \mathrm{A}(y), x.h = z_0, y.h = z_0\}$. Then a rewriting of the query must contain "$\{\langle x, y\rangle \mid \mathrm{A}(x), \mathrm{A}(y), x.g = z_0, y.g = z_0, x.f = x_1, y.f = y_1, \mathrm{A}(x_1), \mathrm{A}(y_1), x_1.h = z_1, y_1.h = z_1\}$", and "$\{\langle x, y\rangle \mid \mathrm{A}(x), \mathrm{A}(y), x.g = z_0, y.g = z_0, x.f = x_1, y.f = y_1, \mathrm{A}(x_1), \mathrm{A}(y_1), x_1.g = z_1, y_1.g = z_1, x_1.f = x_2, y_1.f = y_2, \mathrm{A}(x_2), \mathrm{A}(y_2), x_2.h = z_2, y_2.h = z_2\}$", and so on, which yields a rewriting that is necessarily infinite.[1]

However, if the second form of PFDs is restricted to standard relational FDs of the form "$\mathrm{A} \sqsubseteq \mathrm{A} : f_1, \ldots, f_n \rightarrow f$", then rewriting becomes possible:

8. If $\{x.f = z, y.f = z, \mathrm{A}_1(z), \ldots, \mathrm{A}_k(z)\} \subseteq \psi$ and $\mathrm{A} \sqsubseteq \mathrm{B} : f_1, \ldots, f_n \rightarrow f \in \mathcal{T}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi'\}\}$ for all ψ' of the form $\psi - \{x.f = z, y.f = z, \mathrm{A}_1(z), \ldots, \mathrm{A}_k(z)\} \cup \{\mathrm{A}^x(x), \mathrm{A}^y(y)\} \cup \{\mathrm{B}^x_{1,i_1}(x), \ldots, \mathrm{B}^x_{k,i_k}(x)\} \cup \{\mathrm{B}^y_{1,i_1}(y), \ldots, \mathrm{B}^y_{k,i_k}(y)\} \cup \{x.f_1 = z_1, y.f_1 = z_1, \ldots, x, f_n = z_n, y.f_n = z_n\}\}$, where (a) ($\mathrm{A}^x = \mathrm{A}$ and $\mathrm{A}^y = \mathrm{B}$) or ($\mathrm{A}^x = \mathrm{B}$ and $\mathrm{A}^y = \mathrm{A}$), and (b) ($\mathrm{B}^x_{i,j_i} = \mathrm{B}_{i,j_i}$ and $\mathrm{B}^y_{i,j_i} = \top$) or ($\mathrm{B}^x_{i,j_i} = \top$ and $\mathrm{B}^y_{i,j_i} = \mathrm{B}_{i,j_i}$), for all i and for all combinations of B_{i,i_j} for which $\mathcal{T} \models \forall f.\mathrm{B}_{i,i_j} \sqsubseteq \mathrm{A}_i$ and B_{i,i_j} is maximal w.r.t. $\sqsubseteq$.

A similar adjustment must be performed on the key PFDs to account for equalities implied by FDs and in turn correctly identify ABox individuals (or signal KB inconsistency). Altogether, we have:

Lemma 16 Let $Q = \{\bar{x} \mid \varphi\}$ be a CQ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with standard relational FDs and keys. Then $\bar{a}$ is a certain answer to Q over $\mathcal{K}$ if and only if $\bar{a}$ is in the result of at least one $\{\bar{x} \mid \psi\} \in \mathsf{Fold}_{\mathcal{T}}(Q)$ using reformulations (1-8) over an $\mathcal{A}$ completion using only rules (a,b) and (c.1) in Figure 3 (modified as described above).

3.5 Concrete Classes and Data Types

Although one can treat all primitive concepts uniformly in our approach, in practice one may want to distinguish those that correspond to primitive data types, such as INT and STRING in our running example, since computing a *completion* for these concepts results in collecting all integers or strings occurring in the KB, which is undesirable. To avoid the need for completing these concepts, there are two options, with the first restricting user queries to disallow asking directly about these concepts.[2] Otherwise, to avoid the need for data completion for these concepts, consider that concepts corresponding to data types *should not have outgoing features*, and can therefore be handled using query rewriting:

9. If $\mathrm{A}(x) \in \psi$, A a data type, and $\mathcal{T} \models \mathrm{B} \sqsubseteq \forall f.\mathrm{A}$ then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi - \{\mathrm{A}(x)\} \cup \{\mathrm{B}(y), y.f = x\}\}\}$, where y is a fresh variable.

This additional rule makes it safe to omit completion rules from Figure 3(b) when concepts corresponding to data types are involved.

Example 17 Consider the query $Q = \{\langle x\rangle \mid \mathrm{STRING}(x)\}$. We get reformulations $\{\langle x\rangle \mid \mathrm{PERSON}(y), y.name = x\}$ and $\{\langle x\rangle \mid \mathrm{DEPT}(y), y.name = x\}$ in $\mathsf{Fold}(Q)$, which in turn retrieve all STRINGS recorded in the KB.

The constructions used in Lemmas 12 and 14 (and optionally Lemma 16 and Section 3.5(9)) also yield the upper bounds. In all cases, computing certain answers reduces to evaluating an union of conjunctive queries over a polynomially-sized completion of an ABox. Lower bounds follow from reductions from graph reachability, Horn-SAT, and the DFA intersection problem [Kozen, 1977], respectively. Note that PSPACE-hardness holds even for very simple CQs of the form $\exists x.(A_1(x) \wedge \ldots \wedge A_k(x))$.

Corollary 18 Data complexity for CQ query answering over $\mathcal{CFDI}_{nc}$ KB is in PTIME. CQ query answering over $\mathcal{CFDI}_{nc}$ KB is NLOGSPACE-hard for data complexity (in $|\mathcal{A}|$), PTIME-hard in $|\mathcal{T} + \mathcal{A}|$, and PSPACE-hard for combined complexity.

4 Relational Back-end

Hereon, we assume any non-key PFD occurring in a knowledge base $\mathcal{K}$ is a key or conforms to a standard relational FD. (Recall that this enables our framework to avoid anonymous object invention.) Our objective in this section is to show that this leads to a very practical front-end to existing relational engines in which $\mathcal{K}$ consists of a TBox $\mathcal{T}$ that *embeds* a relational schema $\mathcal{S}$, and a virtual ABox $\mathcal{A}$ that is given by the tuples occurring in relational tables. W.l.o.g., we assume the correspondence is as follows:

1. Primitive concepts (save concepts capturing *concrete values* such as INT and STRING in Figure 1), hereon called *abstract*, correspond to tables, and identities of objects belonging to such concepts are captured by *primary key* values in the tables;
2. Features between abstract concepts, hereon also called *abstract*, are captured by *foreign keys* between the corresponding tables. Note that a single such feature may correspond to a multi-arity foreign key;

[1] This, without resorting to regular path functions as in [Toman and Weddell, 2005] which, however, would preclude the consequent use of an SQL engine.

[2] This is the approach SQL and OQL have taken.

3. Features between abstract concepts and concepts capturing concrete values, hereon called *concrete*, correspond to attributes in the table corresponding to the abstract concept with an appropriate concrete data type, for which UNA is assumed (as is common in SQL).

Note also that non primary key attributes may contain SQL's *null* values to account for data incompleteness.

In our running example, $\mathcal{S}$ consists of "create table" commands that include declarations for primary and foreign keys, such as the following in the case of the CLASS primitive concept for our UNIV TBox:

create table CLASS-TAB (*dname*, *num*, *iname*, *room*, *time*,
primary key (*dname*, *num*),
foreign key (*dname*) to DEPT-TAB,
foreign key (*iname*) to PROF-TAB).

We write $\textsc{Tab}(\text{CLASS})$, $\textsc{Attr}(\text{CLASS})$ and $\textsc{Pk}(\text{CLASS})$ to respectively denote table CLASS-TAB, its set of attributes (*dname*, etc.) and the *sequence* of its primary key attributes (*dname*, *num*). For abstract features, such as $dept$, we use $\textsc{DomAttr}(dept)$, $\textsc{DomTab}(dept)$, and $\textsc{RanTab}(dept)$ to denote the list of source attributes ($dname$), the originating table (CLASS-TAB), and the target table (DEPT-TAB), respectively, in order to describe the associated foreign key.

We assume that $\text{A} \sqsubseteq \neg\text{B} \in \mathcal{T}$ whenever $\textsc{Pk}(\text{A}) \neq \textsc{Pk}(\text{B})$ and that $\textsc{Attr}(\text{B}) \subseteq \textsc{Attr}(\text{A})$ whenever $\mathcal{T} \models \text{A} \sqsubseteq \text{B}$.[3]

With these assumptions, ABox completion *for a consistent knowledge base* will only be required for ABox-TBox interactions (see Figure 3), and is accomplished by repeatedly executing the following updates until no changes occur:[4]

- (*when* $\mathcal{T} \models \text{A} \sqsubseteq \text{B}$)

  $\textsc{Tab}(\text{A}) := \textsc{Tab}(\text{A}) \cup (\textsc{Tab}(\text{A}) \bowtie_{\textsc{Pk}(\text{A})=\textsc{Pk}(\text{B})} \textsc{Tab}(\text{B}))$
  $\textsc{Tab}(\text{B}) := \textsc{Tab}(\text{B}) \cup \pi_{\textsc{Attr}(\text{B})}(\textsc{Tab}(\text{A}))$

- (*when* $\mathcal{T} \models \text{A} \sqsubseteq \forall f.\text{B}$ and f is abstract)

  $\textsc{Tab}(\text{B}) := \textsc{Tab}(\text{B})$
  $\cup\ \pi_{\textsc{Attr}(\text{B})}(\textsc{Tab}(\text{A}) \bowtie_{\textsc{DomAttr}(f)=\textsc{Pk}(\text{B})} \textsc{RanTab}(f))$

- (*when* $\mathcal{T} \models \forall f.\text{B} \sqsubseteq \text{A}$ and f is abstract)

  $\textsc{Tab}(\text{A}) := \textsc{Tab}(\text{A})$
  $\cup\ \pi_{\textsc{Attr}(\text{A})}(\textsc{DomTab}(f) \bowtie_{\textsc{DomAttr}(f)=\textsc{Pk}(\text{B})} \textsc{Tab}(\text{B}))$

As a consequence, this approach effectively equates object identity with primary key values *already occurring in a relational data source*, and therefore *entirely avoids* any need to create Skolem constants that would otherwise be needed for an explicit ABox realization. The final step consists of converting $\mathsf{Fold}(Q)$ to SQL. This is rather clerical task and the main steps consist of replacing

1. concept descriptions by the associated tables;
2. object references by primary keys; and
3. abstract features by corresponding foreign keys.

Finally, to obtain well-formed SQL query, we need to separate the conjuncts to the SELECT, FROM, and WHERE clauses as dictated by the (rather rigid) SQL syntax.

5 Experiments

To confirm practicality, we applied our framework to a modified version of the LUBM benchmark in which we introduce a relational schema. We then defined a knowledge base TBox that embeds this schema and supplied various virtual ABoxes according to a number of configurations of generated data for the LUBM benchmark [Guo *et al.*, 2005]. We performed the tests on MacBook Pro with 3ghz Core i7 CPU, 16gb RAM (OSX 10.11.1), and Postgres 9.4. Figure 4 reports the time needed to load the raw data, to index it, and to compute the ABox completion. The results show that the completion is comparable to raw data loading (a factor of about 2).

LUBM scale	10	50	100	200	500	1000
raw data load	14.9	117.4	234.4	518.1	1341	3034
indexing	10.2	69.7	167.9	377.6	1063	2308
completion	5.9	134.2	286.6	769.5	2452	6853

Figure 4: Experimental Results (in seconds).

Due to space constraints we do not report the individual running times for the LUBM queries. However, all the queries complete within few seconds (typically under a second), even on the largest instances (except for queries 2 and 9 that took 73s and 65s, respectively, on the LUBM 1000 instance). These results are not surprising for reasonably indexed relational instances and can be considered representative of the approach.

6 Summary

The paper presents a novel approach to OBDA over knowledge bases formulated in $\mathcal{CFDI}_{nc}^{\forall-}$, a DL that has been designed to capture relational and object-relational schemata in a natural way. Indeed, there are additional benefits to using $\mathcal{CFDI}_{nc}^{\forall-}$ as the underlying DL:

Example 19 Consider the two queries generated in Example 17. Since the $\mathcal{CFDI}_{nc}^{\forall-}$ TBox allows us to reason about keys and concept disjointness, we can deduce that neither of the two queries needs the `distinct` duplicate elimination in their SQL `select` clauses (since $name$ is a key for both PERSON and DEPT) and, moreover, the union of these two queries can use the `union all` operator (since the two subqueries are disjoint), thus completely avoiding duplicate elimination in the final query.

For more complete development and benefits of these optimizations see [Khizder *et al.*, 2000; Liu *et al.*, 2002; Toman and Weddell, 2011]. There are two additional avenues for extending this proposal: first, the data completion could be performed incrementally and second, the restrictions on *concept compatibility* based on the primary keys of the underlying tables could be relaxed using techniques based on *referring expressions* recently proposed in [Borgida *et al.*, 2016].

[3] This reflects a "structural inheritance" approach to sub-typing in relational schema design that is not strictly necessary since values of inherited attributes for tuples can be computed with additional joins.

[4] We appeal to the relational algebra to express right-hand-sides. Note that attributes in projection operations are assumed to be initialized to null values when missing from argument subexpressions.

References

[Borgida *et al.*, 2016] Alexander Borgida, David Toman, and Grant Weddell. On referring expressions in query answering over first order knowledge bases. In *Principles of Knowledge Representation and Reasoning*, 2016. (in press).

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2015] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Davide Lanti, Martín Rezk, and Guohui Xiao. How to stay ontop of your data: Databases, ontologies and more. In *The Semantic Web: ESWC 2015 Satellite Events - ESWC 2015 Satellite Events Portorož, Slovenia, May 31 - June 4, 2015, Revised Selected Papers*, pages 20–25, 2015.

[Eiter *et al.*, 2012] Thomas Eiter, Magdalena Ortiz, Mantas Simkus, Trung-Kien Tran, and Guohui Xiao. Query rewriting for horn-shiq plus rules. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[Guo *et al.*, 2005] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[Khizder *et al.*, 2000] Vitaliy L. Khizder, David Toman, and Grant Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, pages 1017–1032, 2000.

[Kontchakov *et al.*, 2010] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in DL-Lite. In *Principles of Knowledge Representation and Reasoning*, pages 247–257, 2010.

[Kontchakov *et al.*, 2011] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2656–2661, 2011.

[Kozen, 1977] Dexter Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 254–266. IEEE Computer Society, 1977.

[Liu *et al.*, 2002] Huizhu Liu, David Toman, and Grant Weddell. Fine Grained Information Integration with Description Logic. In *Description Logics 2002*, pages 1–12. CEUR-WS vol.53, 2002.

[Lutz *et al.*, 2009] Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2070–2075, 2009.

[Lutz *et al.*, 2013] Carsten Lutz, Inanç Seylan, David Toman, and Frank Wolter. The combined approach to OBDA: Taming role hierarchies using filters. In *International Semantic Web Conference (1)*, pages 314–330, 2013.

[Toman and Weddell, 2005] David Toman and Grant Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 336(1):181–203, 2005.

[Toman and Weddell, 2009] David Toman and Grant E. Weddell. Applications and extensions of PTIME description logics with functional constraints. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 948–954, 2009.

[Toman and Weddell, 2011] David Toman and Grant E. Weddell. *Fundamentals of Physical Design and Query Compilation*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[Toman and Weddell, 2013] David Toman and Grant E. Weddell. Conjunctive Query Answering in $\mathcal{CFD}_{nc}$: A PTIME Description Logic with Functional Constraints and Disjointness. In *Australasian Conference on Artificial Intelligence*, pages 350–361, 2013.

[Toman and Weddell, 2014] David Toman and Grant E. Weddell. On adding inverse features to the description logic $\mathcal{CFD}^{\forall}_{nc}$. In *PRICAI 2014: Trends in Artificial Intelligence - 13th Pacific Rim International Conference on Artificial Intelligence, Gold Coast, QLD, Australia*, pages 587–599, 2014.

[Toman and Weddell, 2015] David Toman and Grant E. Weddell. On the utility of *CFDI*. In *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015.*, 2015.