Answering Queries over $\mathcal{CFD}_{nc}^{\forall}$ Knowledge Bases

David Toman and **Grant Weddell**

Abstract

We exhibit a procedure in the spirit of OBDA in which a relational engine can be usefully employed to address scalability issues for answering conjunctive queries in the description logic $\mathcal{CFD}_{nc}^{\forall}$, a generalization of the logic $\mathcal{CFD}_{nc}$ in which universal restrictions are permitted on left-hand-sides of inclusion dependencies. The procedure necessarily relies on a combination of the strategies that underlie both the perfect rewriting and combined approaches to OBDA.

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 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 plain data yielding the desired answers (Calvanese et al. 2007), and (ii) the *combined* approaches in which the data is completed using the ontological knowledge (formulated in DL-Lite or $\mathcal{EL}$ logics) in such a way that the original query (modulo ontology-independent filtering) can then be executed over the data completion (Kontchakov et al. 2010; 2011; Lutz et al. 2013; Lutz, Toman, and Wolter 2009).

Recently, Toman and Weddell proposed $\mathcal{CFD}_{nc}$ (Toman and Weddell 2013), a dialect of their $\mathcal{CFD}$ family of DLs (Khizder, Toman, and Weddell 2000; Toman and Weddell 2009) that has PTIME complexity for many of the fundamental reasoning tasks, including computing the certain answers to CQs.

In this paper, we show that CQ answering over $\mathcal{CFD}_{nc}^{\forall}$ knowledge bases[1] 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 $\mathcal{CFD}_{nc}^{\forall}$ 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. We solve the problem by introducing a novel technique based on combining query rewriting with data completion.

Our main technical contributions, in the order presented, are as follows:

- We show that concept satisfiability with respect to a $\mathcal{CFD}_{nc}^{\forall}$ TBox $\mathcal{T}$ is complete for NLOGSPACE by appeal to an automaton that derives from $\mathcal{T}$;
- We exhibit an ABox completion procedure for a given $logic$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with PTIME data complexity, and show how this can be coupled with our automata for checking concept satisfiability to yield a procedure for checking knowledge base consistency, also with PTIME data complexity;
- 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 $\mathcal{K}$.

We also show that the potential for an exponential blowup of the query rewriting cannot be avoided in general since the combined complexity for CQ answering in $\mathcal{CFD}_{nc}^{\forall}$ is PSPACE-complete (unless NP=PSPACE).

We begin in the next section by introducing the syntax and semantics of $\mathcal{CFD}_{nc}^{\forall}$, including a normal form that is assumed in the remainder of the paper. The problem of concept satisfiability for $\mathcal{CFD}_{nc}^{\forall}$ TBoxes is then considered in Section 3. Our second contribution is given in Section 4 in which we present our ABox completion procedure, and then show how consistency of $\mathcal{CFD}_{nc}^{\forall}$ knowledge bases can be

[1] $\mathcal{CFD}_{nc}^{\forall}$ is a generalization of $\mathcal{CFD}_{nc}$ in which value restrictions are also permitted on left-hand-sides of inclusion dependencies; see Section 2 for definitions.

determined in PTIME. Our final contribution relating to the above mentioned mapping of conjunctive queries is the topic of Section 5, and a discussion of related work and summary comments then follow in Sections 6 and 7.

2 The Description Logic $\mathcal{CFD}^{\forall}_{nc}$

$\mathcal{CFD}^{\forall}_{nc}$ is a member of the $\mathcal{CFD}$ family of DLs, all of which are essentially fragments of FO with underlying signatures based on disjoint sets of unary predicate symbols called *primitive concepts*, constant symbols called *individuals* and unary function symbols called *attributes*. Note that incorporating attributes deviates from normal practice to use binary predicate symbols called *roles*. However, attributes make it easier to incorporate concept constructors suited to the capture of relational data sources and constraints such as keys and functional dependencies by a straightforward reification of n-ary predicates. Thus, e.g., a role R in $\mathcal{ALC}$ would correspond to a primitive concept R_C and two attributes *domR* and *ranR* in $\mathcal{CFD}^{\forall}_{nc}$, and an $\mathcal{ALC}$ inclusion dependency $\mathrm{A} \sqsubseteq \forall R.\mathrm{B}$ would be captured as the $\mathcal{CFD}^{\forall}_{nc}$ inclusion dependency $\forall \mathit{domR}.\mathrm{A} \sqsubseteq \forall \mathit{ranR}.\mathrm{B}$.

Definition 1 ($\mathcal{CFD}^{\forall}_{nc}$ Knowledge Bases) Let F, PC and IN be disjoint sets of (names of) attributes, 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 ".". *Concepts* C and D are defined by the grammars on the left-hand-side of Figure 1 in which occurrences of "A" denote primitive concepts. A concept "$\mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$" produced by the last production of the grammar for D is called a *path functional dependency* (PFD). To retain tractability of reasoning (Toman and Weddell 2008), any occurrence of a PFD must also satisfy a *regularity* condition by adhering to one of the following two forms:

$$\begin{array}{ll} (a) & \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \text{ or} \\ (b) & \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.\, f \end{array} \quad (1)$$

A PFD is a *key* if it adheres to the first of these forms.

Metadata and data in a $\mathcal{CFD}^{\forall}_{nc}$ knowledge base $\mathcal{K}$ are respectively defined by a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. Assume $\mathrm{A} \in \mathsf{PC}$, C and D are arbitrary concepts given by the grammars in Figure 1, $\{\mathsf{Pf}_1, \mathsf{Pf}_2\} \subseteq \mathsf{F}^*$ and that $\{a, b\} \subseteq \mathsf{IN}$. Then $\mathcal{T}$ consists of a finite set of *inclusion dependencies* of the form $\mathrm{C} \sqsubseteq \mathrm{D}$, and $\mathcal{A}$ consists of a finite set of facts in form of *concept assertions* $\mathrm{A}(a)$, *basic function assertions* $f(a) = b$ and *path function assertions* $\mathsf{Pf}_1(a) = \mathsf{Pf}_2(b)$. $\mathcal{A}$ is called a *primitive* ABox if it consists only of concept and basic function assertions.

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 $\triangle$, attributes f to be total functions on $\triangle$, and individuals a to be elements of $\triangle$. The interpretation function is extended to path expressions by interpreting id, the empty word, as the identity function $\lambda x.x$, concatenation as function composition, and to derived concept descriptions C or D as defined in Figure 1.

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 $a^{\mathcal{I}} \in \mathrm{A}^{\mathcal{I}}$, a basic function assertion $f(a) = b$ if $f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$ 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 $\mathcal{K}$, and also satisfies UNA if, for any individuals a and b occurring in $\mathcal{K}$, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. □

As usual, allowing conjunction (resp. disjunction) on the right-hand (resp. left-hand) sides of inclusion dependencies is a simple syntactic sugar.

The conditions imposed on PFDs in (1) are necessary to retain PTIME complexity for the reasoning problems (Khizder, Toman, and Weddell 2001; Toman and Weddell 2008) and does not impact the modeling utility of $\mathcal{CFD}^{\forall}_{nc}$ for formatted legacy data sources such as relational databases. It remains possible, for example, to capture arbitrary keys or functional dependencies in a relational schema.

For reasoning tasks, such as TBox and more general KB consistency, it is convenient to assume by default, and without loss of generality, that $\mathcal{CFD}^{\forall}_{nc}$ knowledge bases are given in a normal form.

Lemma 2 (TBox and ABox Normal Forms)
For every $\mathcal{CFD}^{\forall}_{nc}$ TBox $\mathcal{T}$, there exists a conservative extension $\mathcal{T}'$ that adheres to the following (more limited) grammar for $\mathcal{CFD}^{\forall}_{nc}$ concept descriptions.

$$\begin{array}{rcl} \mathrm{C} & ::= & \mathrm{A} \mid \forall f.\mathrm{A} \\ \mathrm{D} & ::= & \mathrm{A} \mid \neg \mathrm{A} \mid \forall f.\mathrm{A} \mid \mathrm{A} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \end{array}$$

Also, for every ABox $\mathcal{A}$, there exists an equivalent ABox $\mathcal{A}'$ containing only assertions of the form $\mathrm{A}(a)$, $f(a) = b$, and $a = b$, where A is a primitive concept. □

Obtaining $\mathcal{T}'$ and $\mathcal{A}'$ from an arbitrary knowledge base $\mathcal{K}$ is achieved by a straightforward introduction of auxiliary names for intermediate concept descriptions and individuals (e.g., see defn. of *simple concepts* in (Toman and Weddell 2008)); the normalized TBox and ABox are linear in the size of the inputs.

3 TBox and Concept Satisfiability

It is easy to see that every $\mathcal{CFD}^{\forall}_{nc}$ TBox $\mathcal{T}$ is consistent (by setting all primitive concepts to be interpreted as the empty set). To test for concept satisfiability, we use the following construction.

Definition 3 (Transition Relation for $\mathcal{T}$) Let $\mathcal{T}$ be a $\mathcal{CFD}^{\forall}_{nc}$ TBox in normal form. We define a transition relation $\delta(\mathcal{T})$ over the set of states

$$S = \mathsf{PC} \cup \{\neg \mathrm{A} \mid \mathrm{A} \in \mathsf{PC}\} \cup \{\forall f.\mathrm{A} \mid \mathrm{A} \in \mathsf{PC}, f \in \mathsf{F}\}$$

and the alphabet F as follows:

$$\begin{array}{rl} \mathrm{C} \xrightarrow{\epsilon} \mathrm{D} \in \delta(\mathcal{T}) & \text{if} \quad \mathrm{C} \sqsubseteq \mathrm{D} \in \mathcal{T} \\ \forall f.\mathrm{A} \xrightarrow{f} \mathrm{A} \in \delta(\mathcal{T}) & \end{array}$$

where ϵ is the empty letter transition, $f \in \mathsf{F}$, $A \in \mathsf{PC}$, and $C, D \in S$. □

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

Figure 1: $\mathcal{CFD}_{nc}^{\forall}$ concepts.

The transition relation allows us to construct *nondeterministic finite automata* (NFA) that can be used for various reasoning problems that relate to a $\mathcal{CFD}_{nc}^{\forall}$ TBox $\mathcal{T}$. Note that we follow common practice in automata theory and use ϵ for the empty letter in transition relations.[2]

Lemma 4 Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall}$ TBox, C a primitive concept, and D a primitive concept or the complement of a primitive concept.. We define $M = (S, \{\mathrm{C}\}, \{\mathrm{D}\}, \delta(\mathcal{T}))$ to be an NFA with the set of states S (as above), start state C, final state D, and transition relation $\delta(\mathcal{T})$. Then $\mathcal{T} \models \mathrm{C} \sqsubseteq \forall \mathsf{Pf}.\mathrm{D}$ whenever $\mathsf{Pf} \in \mathcal{L}(M)$.

Proof (sketch) For $\mathsf{Pf} \in \mathcal{L}(M)$ there must be a run

$$\mathrm{C} = \mathrm{A}_0 \xrightarrow{l_1} \mathrm{A}_1 \xrightarrow{l_2} \mathrm{A}_2 \cdots \mathrm{A}_{k-1} \xrightarrow{l_k} \mathrm{A}_k = \mathrm{D}$$

in M where $l_i \in \mathsf{F} \cup \{\epsilon\}$ and such that $\mathsf{Pf} = l_1.l_2.\cdots.l_k$. It follows from the definition of $\delta(\mathcal{T})$ that $\mathrm{A}_{i-1} \xrightarrow{l_i} \mathrm{A}_i$ exists if $\mathrm{A}_{i-1} \sqsubseteq \mathrm{A}_i$, for $l_i = \epsilon$, or $l_i \in \mathsf{F}$ (and hence these dependencies are trivially implied by $\mathcal{T}$). The claim then follows by simple transitive reasoning, all necessary cases derive from the fact that

$$\{\mathrm{B}_1 \sqsubseteq \forall \mathsf{Pf}.\mathrm{B}_2, \mathrm{B}_2 \sqsubseteq \forall \mathsf{Pf}'.\mathrm{B}_3\} \models \mathrm{B}_1 \sqsubseteq \forall \mathsf{Pf}.\mathsf{Pf}'.\mathrm{B}_3,$$

and the lemma then follows by induction on the length of the run. □

Concept Satisfiability

The problem of *primitive concept satisfiability* is to determine, for a given concept A and TBox $\mathcal{T}$, if there exists an interpretation $\mathcal{I}$ for $\mathcal{T}$ in which $\mathrm{A}^{\mathcal{I}}$ is non-empty. Given a primitive concept A and TBox $\mathcal{T}$, one can test for primitive concept satisfiability by using the NFA, denoted $\mathsf{nfa}_{\mathrm{B}}^{\mathrm{A}}(\mathcal{T})$:

$$(S, \{\mathrm{A}\}, \{\mathrm{B}\}, \delta(\mathcal{T})), \tag{2}$$

with states induced by primitive concepts, by their complements, and by value restrictions, with start state A, with the set of final states $\{\mathrm{B}\} \subseteq S$, and with transition relation $\delta(\mathcal{T})$.

Theorem 5 (Primitive Concept Satisfiability) A is satisfiable with respect to the TBox $\mathcal{T}$ if and only if

$$\mathcal{L}(\mathsf{nfa}_{\mathrm{B}}^{\mathrm{A}}(\mathcal{T}) \cap \mathcal{L}(\mathsf{nfa}_{\neg\mathrm{B}}^{\mathrm{A}}(\mathcal{T}) = \emptyset$$

for every $\mathrm{B} \in \mathsf{PC}$.

Proof (sketch) Assume A is non-empty and hence there is $a \in \mathrm{A}^{\mathcal{I}}$. For a primitive concept $\mathrm{B} \in \mathsf{PC}$, a word Pf in the intersection language of the two automata above is a witness of the fact that $\mathsf{Pf}^{\mathcal{I}}(a^{\mathcal{I}}) \in \mathrm{B}^{\mathcal{I}}$ and $\mathsf{Pf}^{\mathcal{I}}(a^{\mathcal{I}}) \in \neg\mathrm{B}^{\mathcal{I}}$ must hold in every model of $\mathcal{T}$, for reasons analogous to the proof of Lemma 4, which leads to a contradiction since Pf is a (total) function.

Conversely, if no such word exists then one can construct a *deterministic* finite automaton from $\mathsf{nfa}_{\mathrm{B}}^{\mathrm{A}}(\mathcal{T})$, using the standard subset construction, in which no state containing both B and $\neg$B is reachable from the start state A. Unfolding the transition relation of this automaton, starting from the state A, labeling nodes by the concepts associated with the automaton's states, and adding missing features to complete trees in which no primitive concept is true for any node, yields a tree interpretation that satisfies $\mathcal{T}$ (in particular in which all PFD constraints are satisfied vacuously) and whose root provides a witness for satisfiability of A (as we can simply assert $a \in \mathrm{A}$). □

To test for emptiness of (2), we use an graph connectivity algorithm that (nondeterministically) searches for a path from (A, A) to $(\mathrm{B}, \neg\mathrm{B})$ in the (virtual) poly-sized product automaton (Hopcroft and Ullman 1979); the following result is then immediate.

Corollary 6 Concept satisfiability with respect to $\mathcal{CFD}_{nc}^{\forall}$ TBoxes is complete for NLOGSPACE.

Note that this procedure can be trivially extended to test for consistency of *conjunctions of concepts and their negations* in $\mathcal{CFD}_{nc}^{\forall}$: to test for satisfiability of $\mathrm{B}_1 \sqcap \ldots \sqcap \mathrm{B}_k$ with respect to $\mathcal{T}$ we simply introduce a fresh primitive concept A and test for satisfiability of A in $\mathcal{T} \cup \{\mathrm{A} \sqsubseteq \mathrm{B}_i \mid i \leq k\}$. It is, however, impossible to *precompute* all such inconsistent concepts since this would require consideration of all possible *types* over PC (or finite subsets of primitive concepts), a process essentially equivalent to constructing an equivalent deterministic automaton which can require exponential time (Hopcroft and Ullman 1979).

4 ABox Completion

To test for consistency we follow the path first outlined for the *combined approach* to CQ answering in PTIME-

[2] Another option would have been to use id for this purpose, but we thought, on balance, that this would hinder readability.

if	$a = b, b = c \in \mathcal{A}$	then add	$a = c$	to $\mathcal{A}$
if	$f(a) = b, b = c \in \mathcal{A}$	then add	$f(a) = c$	to $\mathcal{A}$
if	$a = b, f(b) = c \in \mathcal{A}$	then add	$f(a) = c$	to $\mathcal{A}$
if	$f(a) = b, f(a) = c \in \mathcal{A}$	then add	$b = c$	to $\mathcal{A}$
if	$a = b, \mathrm{A}(a) \in \mathcal{A}$	then add	$\mathrm{A}(b)$	to $\mathcal{A}$

ABox Equality Interactions

if $\mathrm{A}(a) \in \mathcal{A}$	and $\epsilon \in \mathcal{L}(\mathsf{nfa}_{\mathrm{B}}^{\mathrm{A}}(\delta(\mathcal{T})))$	then add	$\mathrm{B}(a)$ to $\mathcal{A}$
if $\mathrm{A}(a), f(a) = b \in \mathcal{A}$	and $f \in \mathcal{L}(\mathsf{nfa}_{\mathrm{B}}^{\mathrm{A}}(\delta(\mathcal{T})))$	then add	$\mathrm{B}(b)$ to $\mathcal{A}$
if $\mathrm{A}(b), f(a) = b \in \mathcal{A}$	and $\epsilon \in \mathcal{L}(\mathsf{nfa}_{\mathrm{B}}^{\forall f.\mathrm{A}}(\delta(\mathcal{T})))$	then add	$\mathrm{B}(a)$ to $\mathcal{A}$

ABox–$\delta(\mathcal{T})$ Interactions

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

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

ABox–PFD Interactions

Figure 2: ABox Completion Rules.

complete DLs (Lutz, Toman, and Wolter 2009) by *completing* the explicit data using the TBox. Note for our case, however, that we do not attempt to generate *auxiliary anonymous individuals* to satisfy totality of features (the counterpart of qualified existential restrictions in $\mathcal{EL}$) since there can be exponentially many such individuals with distinct concept membership.

Example 7 Consider a $\mathcal{CFD}_{nc}^{\forall}$ 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, Toman, and Wolter 2009).

Hence, our ABox completion focuses exclusively on concept membership of existing individuals and on realizing equational constraints mandated by the PFDs. The later involves equating anonymous individuals at most one feature "away" from an ABox individual (see "ABox-PFD Interactions in Figure 2) and thus only linearly many new individuals are needed.

Definition 8 (ABox Completion) The *completion of ABox $\mathcal{A}$ with respect to $\mathcal{T}$*, denoted $\mathsf{completion}_{\mathcal{T}}(\mathcal{A})$, is the least ABox that contains $\mathcal{A}$ and is closed under the rules in Figure 2.

Lemma 9 Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFD}_{nc}^{\forall}$ knowledge base such that $\mathrm{A}(a) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})$. Then $\mathcal{K} \models \mathrm{A}(a)$.

Proof (sketch) The completion rules in Figure 2 only add facts implied by $\mathcal{K}$.

Note that the converse is contingent on consistency of $\mathcal{K}$.

Knowledge Base Consistency

The automata-based approach to *concept satisfiability* can be applied to the more general problem of knowledge base consistency.

Theorem 10 (Knowledge Base Consistency)
Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFD}_{nc}^{\forall}$ knowledge base. $\mathcal{K}$ is consistent if and only if $\{\mathrm{A} \mid \mathrm{A}(a) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})\}$ is satisfiable with respect to $\mathcal{T}$ for every individual a in $\mathsf{completion}_{\mathcal{T}}(\mathcal{A})$.

Proof (sketch) If $\{\mathrm{A} \mid \mathrm{A}(a) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})\}$ is not satisfiable for some a then the knowledge base $(\mathcal{T}, \{\mathrm{A}(a) \mid \mathrm{A}(a) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})\})$ is inconsistent and hence $\mathcal{K}$ is also inconsistent due to Lemma 9.

For the other direction, construct an interpretation $\mathcal{I}$ for $\mathcal{K}$ as follows: begin by closing $\mathcal{A}$ under the rules in Figure 2 and then proceed by extending the closure with anonymous objects by unfolding $\delta(\mathcal{T})$ for every individual that does not satisfy the totality of features requirement. It is then easy to show that $\mathcal{I} \models \mathcal{T}$. □

Note that the ABox individuals are considered *separately* (i.e., without considering the ABox equalities); however, were an inconsistency with respect to an individual forced by traversing a feature within the ABox, this inconsistency will be detected when the target individual is considered in the above theorem.

The construction yields a PTIME algorithm for consistency checking, the lower bound for $\mathcal{CFD}_{nc}$ has been already established (Toman and Weddell 2013).

Corollary 11 Knowledge base consistency for $\mathcal{CFD}_{nc}^{\forall}$ is PTIME-complete. □

The interpretation constructed in the *only if* part of the proof is called the *canonical model* of $\mathcal{K}$ and is analogous to the minimal models of horn theories.

5 Conjunctive Queries over $\mathcal{CFD}^{\forall}_{nc}$ KBs

We now show that CQ answering is tractable in data complexity for $\mathcal{CFD}^{\forall}_{nc}$. Note that this also makes the problem of evaluating instance queries tractable since CQ answering subsumes instance query evaluation.

A *conjunctive query* (CQ) is an expression of the form $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\}$ where BODY is a conjunction of atomic formulas of the form $\mathrm{C}(x)$ and $\mathsf{Pf}(x) = \mathsf{Pf}'(y)$ for C a $\mathcal{CFD}^{\forall}_{nc}$ concept description not containing PFDs, $\mathsf{Pf}, \mathsf{Pf}' \in \mathsf{F}^*$, and x are variables among $\bar{x} \cup \bar{y}$. We often conflate the BODY of the query with the set of its atomic conjuncts. We call the variables $\bar{x}$ the *answer variables*. A *union of CQ* (UCQ) is a set of CQ that denotes a disjunction of the formulas that define the individual CQs. An *answer to a CQ* φ w.r.t. a KB $\mathcal{K}$ is a vector of individuals $\bar{a} \subseteq \mathsf{IN}$ such that $\mathcal{K} \models \varphi(\bar{a})$ where $\varphi(\bar{a})$ is a formula obtained from φ by substituting $\bar{x}$ by $\bar{a}$. We assume that CQs are connected; otherwise we simply process each component separately.

Without loss of generality we can assume that all CQs are in normal form: the concepts used in the CQs are primitive concepts or their negations and the equational atoms of the form $f(x) = y$. It is easy to see that every CQ can be transformed to a equivalent one by introducing additional variables and existential quantifiers.

To compute answers for a CQ φ we use the notion of *CQ folding*. We need the following auxiliary definition:

Definition 12 Let $\mathcal{T}$ be a $\mathcal{CFD}^{\forall}_{nc}$ TBox and C a primitive concept or the complement of a primitive concept. We say that a primitive concept A is a Pf-precondition of C in $\mathcal{T}$ if $\mathsf{Pf} \in \mathcal{L}(\mathsf{nfa}^{A}_{C}(\delta(\mathcal{T})))$.

Lemma 13 Let $\mathcal{T}$ be a $\mathcal{CFD}^{\forall}_{nc}$ TBox and $\mathrm{A}_1, \ldots, \mathrm{A}_k$ all Pf-preconditions of C in $\mathcal{T}$. Then, in every model I of $\mathcal{T}$, $o \in \bigcup_{0<i\leq k} \mathrm{A}_i^{\mathcal{I}}$ implies $\mathsf{Pf}^{\mathcal{I}}(o) \in \mathrm{C}^{\mathcal{I}}$ for $o \in \triangle$.

Proof (sketch) Follows immediately from Lemma 4.

The above enables us to replace concepts of the form $\neg A$ and $\forall f.A$ by their preconditions w.r.t. $\mathcal{T}$, thus ensuring each of the queries will be in normal form.

Definition 14 Let φ be a CQ. We define a set $\mathsf{Fold}_{\mathcal{T}}(\varphi)$ with respect to $\mathcal{T}$ to be the least set of CQ that contains φ and is closed under the following rules.

1. If $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$, $\{\neg \mathrm{A}(x)\} \subseteq \text{BODY}$ then
$$\{\bar{x} \mid \exists \bar{y}.\text{BODY}\} - \{\neg \mathrm{A}(x)\} \cup \{\mathrm{B_i}(x)\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$$
for all $\mathrm{B_i}$ an ϵ-precondition of $\neg \mathrm{A}$;
2. If $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$, $\{f(x) = y, \mathrm{A}_1(y), \ldots, \mathrm{A_k}(y)\} \subseteq \text{BODY}$, and y does not appear elsewhere in BODY nor in $\bar{x}$, then
$$\{\bar{x} \mid \exists \bar{y} - \{y\}.(\text{BODY} - \{f(x) = y, \mathrm{A}_1(y), \ldots, \mathrm{A_k}(y)\} \cup \{\mathrm{B}_1^{\mathrm{j}_1}(x), \ldots, \mathrm{B}_{\mathrm{k}}^{\mathrm{j}_k}(x)\})\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$$
for all possible combinations of concepts $\mathrm{B}_{\mathrm{i}}^{\mathrm{j_i}}$ that are f-preconditions of $\mathrm{A_i}$ w.r.t. $\mathcal{T}$.
3. If $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$ and $\{f(x) = y, f(x') = y\} \subseteq \text{BODY}$, then $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\}[x/x'] \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$;

The intuition that underlies this definition is that, to find query answers, it is now sufficient to *match* the queries in $\mathsf{Fold}(\varphi)$ explicitly against the (extended) ABox (cf. Definition 8) and verify correct concept membership for these nodes as prescribed by the query since possible matches outside of this ABox are reduced to primitive membership checks against Pf-preconditions.

Lemma 15 Let φ be a CQ with at least one answer variable. Then $\bar{a}$ is an answer to φ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if and only if there is a mapping $\mu : \bar{x} \cup \bar{y} \to \mathsf{IN}$ to the set of ABox individuals in $\mathsf{completion}_{\mathcal{T}}(\mathcal{A})$, such that

1. $\mu(x)$ is an individual in $\mathcal{A}$ for $x \in \bar{x}$ an answer variable;
2. $f(\mu(x)) = \mu(y) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})$ for all $f(x) = y \in \text{BODY}$; and
3. $\mathrm{A}(\mu(x)) \in \mathsf{completion}_{\mathcal{T}}(\mathcal{A})$ for all $\mathrm{A}(x) \in \text{BODY}$,

for at least one $\{\bar{x} \mid \exists \bar{y}.\text{BODY}\} \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$.

Proof (sketch) Observing that the extended ABox is essentially a part of the minimal model of $\mathcal{K}$ (since $\mathcal{K}$ is Horn) and that every element of $\mathsf{Fold}(\varphi)$ implies φ, it is easy to see that whenever (1-3) are satisfied, there is a match of φ in the minimal model and thus $\bar{a}$ is an answer. Conversely, if a match of φ 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 one variable must be bound to an ABox individual) and the reminder of the match must be forest-like. Hence, one of the queries in $\mathsf{Fold}_{\mathcal{T}}(\varphi)$ matches in the ABox making the remaining conjuncts implied by $\mathcal{T}$ due to Lemma 13. □

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 the canonical model of $\mathcal{K}$. To achieve this, we determine every primitive type $\{\mathrm{A}_1, \ldots, \mathrm{A}_n\} \subseteq \mathsf{PC}$ (of a potential completed ABox individual) whether C must be realized in the canonical model in which an extended ABox individual belongs to such a primitive type. We use the following construction: Let $\mathcal{T}$ be a $\mathcal{CFD}^{\forall}_{nc}$ TBox, $\{\mathrm{A}_1, \ldots, \mathrm{A}_n\} \subseteq \mathsf{PC}$ a consistent primitive type w.r.t. $\mathcal{T}$, and assume the following query:

$$\psi = \{\emptyset \mid \exists y.\mathrm{B}_1(y) \wedge \ldots \wedge \mathrm{B}_k(y)\}^3.$$

We define an automaton

$$M(\psi) = \mathsf{nfa}^{\mathrm{A}_0}_{\mathrm{B}_1}(\delta) \times \ldots \times \mathsf{nfa}^{\mathrm{A}_0}_{\mathrm{B}_k}(\delta)$$

where $\delta = \delta(\mathcal{T}) \cup \{A_0 \xrightarrow{\epsilon} \mathrm{A}_1, \ldots, A_0 \xrightarrow{\epsilon} \mathrm{A}_n\}$ and A_0 is a primitive concept not occurring in $\mathcal{T}$. This construction also indirectly yields a PSPACE lower bound on combined complexity of query answering in $\mathcal{CFD}^{\forall}_{nc}$ by reduction from the DFA intersection problem (Garey and Johnson 1979).

[3] As a consequence of the definition of $\mathsf{Fold}_{\mathcal{T}}$, it suffices to consider only queries of this form since more complex queries are simplified by the folding process.

Definition 16 Let $\mathcal{T}$ be a $\mathcal{CFD}^{\forall}_{nc}$ TBox, $\{\mathrm{A}_1, \ldots, \mathrm{A}_n\} \subseteq \mathsf{PC}$ a primitive type, and $\psi \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$ of the form $\{\emptyset \mid \exists y.\mathrm{B}_1(y), \ldots, \mathrm{B}_k(y)\}$. We say that the type $\{\mathrm{A}_1, \ldots, \mathrm{A}_n\}$ forces ψ if $M(\psi)$ is nonempty.

Now, whenever such a query ψ appears in $\mathsf{Fold}_{\mathcal{T}}(\varphi)$, we add $\exists x.T(x)$ for all T that force ψ w.r.t. $\mathcal{T}$. These additional queries can be evaluated solely with respect to the completed ABox and guarantee that ψ is realized outside of the ABox whenever a match is found.

Theorem 17 Let φ be a CQ with at least one answer variable. Then $\bar{a}$ is an answer to φ over $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if and only if $\mathsf{completion}_{\mathcal{T}}(\mathcal{A}) \models \psi(\bar{a})$ for at least one $\psi \in \mathsf{Fold}_{\mathcal{T}}(\varphi)$.

Proof (sketch) Follows from Lemma 15 and the observation that closed queries that correspond to $\mathcal{CFD}^{\forall}_{nc}$ concepts are handled using the construction in Definition 16.

Analyzing the above constructions, it is easy to verify that overall data complexity of CQ answering over $\mathcal{CFD}^{\forall}_{nc}$ knowledge bases is in PTIME: an ABox completion can be realized by a Datalog program that depends only on $\mathcal{T}$, and a subsequent evaluation of a UCQ (defined by $\mathsf{Fold}_{\mathcal{T}}(\varphi)$) is in AC_0. This matches the lower bound established for $\mathcal{CFD}_{nc}$ in (Toman and Weddell 2013) that precludes using perfect rewriting alone for CQ answering. Also note that $|\mathsf{Fold}_{\mathcal{T}}(\varphi)|$ is worst-case exponential in $|\mathcal{T}| + |\varphi|$; this, however, is unavoidable as the combined complexity of CQ answering is hard for PSPACE even for $\mathcal{CFD}_{nc}$ while merely NP-complete for UCQ. Thus, unless NP=PSPACE, the blowup cannot be avoided. This observation also precludes the various *filtering* approaches used previously for the combined approach (Kontchakov et al. 2010; 2011; Lutz et al. 2013; Lutz, Toman, and Wolter 2009). Hence the combination of the combined approach and perfect rewriting is necessary for CQ answering over $\mathcal{CFD}^{\forall}_{nc}$.

6 Related Work

An early version of the $\mathcal{CFD}$ dialect first appeared in (Khizder, Toman, and Weddell 2000); the name was a contraction of "CLASSIC with FDs". The present form appears in (Toman and Weddell 2009), which also explored the consequences of adding additional concept constructors on the complexity of concept subsumption problems. Dialect $\mathcal{CFD}_{nc}$ was recently proposed as a modification of $\mathcal{CFD}$ in (Toman and Weddell 2013), mainly to gain PTIME data complexity for conjunctive query answering. However, scalable OBDA in the sense we have outlined was not possible with the query answering approach used in this work.

In (Toman and Weddell 2009), the authors outline an alternative approach to computing the certain answers to so-called *attribute connected* conjunctive queries for the logic $\mathcal{CFD}$. The approach reduces such problems to concept subsumption problems in which path agreements are used in posed question concepts to encode an ABox.

Scalable OBDA based on a perfect rewriting of conjunctive queries was developed for DL-Lite (Calvanese et al. 2007). Note however, that if perfect rewriting suffices to accomplish scalable OBDA for a DL dialect, the underlying ontology must be limited to logics without (general) qualified value or existential restrictions. The combined approach to scalable OBDA, first shown for a member of the $\mathcal{EL}$ family of logics (Lutz, Toman, and Wolter 2009), avoids this restriction. The combined approach can also be applied to the DL-Lite family (Kontchakov et al. 2010), even in the presence of role hierarchies (Lutz et al. 2013). However, for the reasons outlined in this paper, the underlying logics cannot support (even limited) use of *value restrictions*.

7 Summary

We have introduced a new member of the $\mathcal{CFD}$ family of DL dialects called $\mathcal{CFD}^{\forall}_{nc}$ with the following notable properties.

- $\mathcal{CFD}^{\forall}_{nc}$ is a generalization of $\mathcal{CFD}_{nc}$. Consequently, it inherits the ability of $\mathcal{CFD}_{nc}$ to capture terminological cycles with universal restrictions over functional roles, to capture a rich variety of functional constraints over functional role paths, and to express disjointness of atomic concepts. In addition, it is now possible in $\mathcal{CFD}^{\forall}_{nc}$ to have universal restrictions occurring on left-hand-sides of inclusion dependencies.
- We have established the foundations for scalable OBDA for $\mathcal{CFD}^{\forall}_{nc}$ knowledge bases for which both knowledge base consistency and CQ answering have PTIME data complexity. In particular, we show how CQ answering over $\mathcal{CFD}^{\forall}_{nc}$ knowledge bases can be reduced to the computation of a data completion that can then be stored in a RDBMS, followed by an execution of relational queries over the completion.

We have also shown that it is *not* possible for scalable OBDA over a $\mathcal{CFD}^{\forall}_{nc}$ knowledge base $\mathcal{K}$ that is based exclusively on either a perfect rewriting approach or on a combined approach. Moreover, we have shown that (under common complexity-theoretic assumptions) the rewritten CQ is necessarily worst-case exponential in the size of the original query. However, work on (1) reducing the complexity of generated queries in perfect rewriting approaches (Rosati and Almatelli 2010), e.g., removing members of a union of conjunctive queries that are subsumed by other members, and on (2) using interval encoding to compress the expansion of an ABox in combined approaches (Agrawal, Borgida, and Jagadish 1989; Rodriguez-Muro and Calvanese 2012) can also be applied in our setting.

The complexity landscape of most of the variants of $\mathcal{CFD}$ have now been resolved. In particular, see (Toman and Weddell 2014) in the case of $\mathcal{CFD}_{nc}$ and (Toman and Weddell 2009) for the case of $\mathcal{CFD}$ (which allows conjunction on left-hand-sides of inclusion dependencies, but disallows negation on right-hand-sides).

There are additional issues relating to $\mathcal{CFD}^{\forall}_{nc}$ that merit further investigation. First, it is desirable to incorporate limited forms of equational constraints while preserving tractability of reasoning. And second, although the modeling utility is unclear, the consequences of allowing PFDs on the left-hand-sides of inclusion dependencies in $\mathcal{CFD}^{\forall}_{nc}$ is still open.

References

Agrawal, R.; Borgida, A.; and Jagadish, H. V. 1989. Efficient management of transitive relationships in large data and knowledge bases. In Clifford, J.; Lindsay, B. G.; and Maier, D., eds., *SIGMOD Conference*, 253–262. ACM Press.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning* 39(3):385–429.

Garey, M. R., and Johnson, D. S. 1979. *Computers and intractability*. Freeman.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Khizder, V. L.; Toman, D.; and Weddell, G. 2000. Reasoning about Duplicate Elimination with Description Logic. In *Rules and Objects in Databases (DOOD, part of CL'00)*, 1017–1032.

Khizder, V. L.; Toman, D.; and Weddell, G. 2001. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *Int. Conf. on Database Theory ICDT'01*, 54–67.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2010. The combined approach to query answering in DL-Lite. In *Principles of Knowledge Representation and Reasoning*.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2011. The combined approach to ontology-based data access. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2656–2661.

Lutz, C.; Seylan, I.; Toman, D.; and Wolter, F. 2013. The combined approach to OBDA: Taming role hierarchies using filters. In *International Semantic Web Conference (1)*, 314–330.

Lutz, C.; Toman, D.; and Wolter, F. 2009. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2070–2075.

Rodriguez-Muro, M., and Calvanese, D. 2012. High performance query answering over DL-Lite ontologies. In Brewka, G.; Eiter, T.; and McIlraith, S. A., eds., *KR*. AAAI Press.

Rosati, R., and Almatelli, A. 2010. Improving query answering over DL-Lite ontologies. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *KR*. AAAI Press.

Toman, D., and Weddell, G. E. 2008. On keys and functional dependencies as first-class citizens in description logics. *J. Autom. Reasoning* 40(2-3):117–132.

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

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

Toman, D., and Weddell, G. E. 2014. Pushing the $\mathcal{CFD}_{nc}$ Envelope. In *Proc. International Workshop on Description Logics DL2014*, 340–351.