Pushing the $\mathcal{CFD}_{nc}$ Envelope

David Toman and Grant Weddell

Cheriton School of Computer Science
University of Waterloo, Canada
{david,gweddell}@cs.uwaterloo.ca

Abstract. We consider the consequences on basic reasoning problems of allowing negated primitive concepts on left-hand-sides of inclusion dependencies in the description logic dialect $\mathcal{CFD}_{nc}$. Although earlier work has shown that this makes CQ answering coNP-complete, we show that TBox consistency and concept satisfiability remain in PTIME. We also show that knowledge base consistency and instance retrieval remain in PTIME if a $\mathcal{CFD}_{nc}$ knowledge base satisfies a number of additional conditions, and that failing any one of these conditions will alone lead to intractability for these problems.

1 Introduction

Dialects of *description logic* (DLs) with PTIME complexity of reasoning services have become an important tool in addressing scalability issues in the semantic web, in particular with regard to SPARQL query evaluation in the context of the OWL 2 direct semantics entailment regime. Indeed, the W3C has adopted two DL fragments of OWL 2 that are designed to ensure PTIME data complexity for the key services of checking for *knowledge base* (KB) consistency and for *conjunctive query* (CQ) answering. Called *profiles*, the DLs are $\mathcal{EL}$++ [2] and DL-Lite [1, 4]. Medical ontologies are an important factor in the design of $\mathcal{EL}$++ while DL-Lite is designed to address access to legacy relational data sources in cases where the underlying relational schema was derived via ER modeling.

Toman and Weddell have proposed an alternative $\mathcal{CFD}$ family of DLs with PTIME complexity of various reasoning services [8, 14, 15] and that were also designed to address the problem of access to legacy relational data sources. The CFD family is, however, incomparable to the other logics in terms of expressive power: its focus was to address at least the common varieties of data dependencies in legacy relational data sources, in particular, arbitrary collections of primary keys, unary foreign keys and functional dependencies.

One member of this family, $\mathcal{CFD}_{nc}$, has PTIME complexity with respect to the size of a knowledge base for checking KB consistency, and PTIME data complexity for CQ answering [15]. Notably, $\mathcal{CFD}_{nc}$ retains the ability to capture data dependencies that are common to the $\mathcal{CFD}$ family: support for terminological cycles with universal restrictions over functional roles, and the ability to capture a rich variety of functional constraints over functional role paths.

In this paper, we consider the consequences of relaxing a number of syntactic restrictions in $\mathcal{CFD}_{nc}$ on KB consistency checking and other reasoning tasks. In particular, we consider the DL $\mathcal{CFD}_{nc}^{\forall,\neg}$ which now allows value restrictions and negated primitive concepts to occur on the left-hand-sides of inclusion dependencies that comprise the "metadata", called a TBox, of a given $\mathcal{CFD}_{nc}^{\forall,\neg}$ KB. Our results, in the order established, are as follows.

- We show that TBox consistency for $\mathcal{CFD}_{nc}^{\forall,\neg}$ is no longer trivial, unlike the case for $\mathcal{CFD}_{nc}$ and many other lightweight logics, but remains in NLOGSPACE.
- We show that $\mathcal{CFD}_{nc}^{\forall,\neg}$ KB consistency is NP-complete and identify precise fragments of $\mathcal{CFD}_{nc}^{\forall,\neg}$ where KB consistency remains tractable. Interestingly, when negated concepts are allowed on the left-hand side of inclusion dependencies, this boundary is tied to the structure of keys and to the *unique name assumption* (UNA).

We begin in the next section by introducing the syntax and semantics of $\mathcal{CFD}_{nc}^{\forall,\neg}$, and talk about some of its key features and limitations. In Section 3, we consider the problem of TBox consistency, and then more general KB consistency in Section 4. A review of related work and summary comments then follows in Section 5.

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

A formal definition of $\mathcal{CFD}_{nc}^{\forall,\neg}$ knowledge bases and the above reasoning problems now follows. Observe that the logic is based on *attributes* or *features* denoting unary functions instead of the more common case of *roles* denoting arbitrary binary relations. However, this is not really an issue; $\mathcal{CFD}_{nc}^{\forall,\neg}$ is ideal for expressing reification for predicates of any arity (see comments following).

Definition 1 ($\mathcal{CFD}_{nc}^{\forall,\neg}$ 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 ".". *Concept descriptions* 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 avoid undecidability [13], any occurrence of a PFD must also satisfy a *regularity* condition by adhering to one of the following two forms:

$$\begin{array}{l} 1.\ \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \ \text{ or} \\ 2.\ \mathrm{C} : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.f \end{array} \qquad (1)$$

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

Metadata and data in a $\mathcal{CFD}_{nc}^{\forall,\neg}$ 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*

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

Fig. 1. $\mathcal{CFD}_{nc}^{\forall,\neg}$ Concepts.

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) distinguish, e.g., PFDs of the form $\mathrm{C} : f \to id$ and $\mathrm{C} : f \to g$ from PFDs of the form $\mathrm{C} : f \to g.h$, and are necessary to retain PTIME complexity for reasoning problems [9, 13]. However, this does not impact the modelling utility of $\mathcal{CFD}_{nc}^{\forall,\neg}$ for formatted legacy data sources. In particular, it remains possible to capture arbitrary keys or functional dependencies in a relational schema.

The grammar for $\mathcal{CFD}_{nc}^{\forall,\neg}$, unlike $\mathcal{CFD}_{nc}$, allows an unrestricted use of concept constructors for value restrictions and concept negation on left-hand-sides of inclusion dependencies. To see that this constitutes a genuine enhancement of modelling utility, consider an *enrollment* relation for a hypothetical relational data source about students and courses. In a $\mathcal{CFD}_{nc}$ TBox, the relation would be reified as the ENROLLMENT primitive concept with *Student* and *Course* features "typed" as follows:

$$\text{ENROLLMENT} \sqsubseteq (\forall \mathit{Student}.\text{STUDENT}) \sqcap (\forall \mathit{Course}.\text{COURSE}).$$

With the added flexibility of value restrictions in $\mathcal{CFD}_{nc}^{\forall,\neg}$, it is now straightforward to further restrict who may enrol in graduate courses:

$$\forall \mathit{Course}.\text{GRADCOURSE} \sqsubseteq \forall \mathit{Student}.\text{GRADSTUDENT}.$$

The added flexibility relating to negation can be used, e.g., to introduce a "top" concept, say THING, by asserting the following:

$$\text{STUDENT} \sqsubseteq \text{THING} \qquad \neg\text{STUDENT} \sqsubseteq \text{THING}.$$

Such a concept can then be used, e.g., to enforce general range restrictions for attributes:

$$\text{THING} \sqsubseteq \forall \mathit{Student}.\text{STUDENT}.$$

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

Lemma 2 ($\mathcal{CFD}_{nc}^{\forall,\neg}$ KB Normal Form) For every KB $(\mathcal{T}, \mathcal{A})$, there is an equi-satisfiable KB $(\mathcal{T}', \mathcal{A}')$ in which $\mathcal{T}'$ adheres to the following (more limited) grammar for $\mathcal{CFD}_{nc}^{\forall,\neg}$ concept descriptions:

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

and also where ABox $\mathcal{A}'$ contains only assertions of the form $f(a) = b$ and $a = b$. □

Obtaining $\mathcal{T}'$ and $\mathcal{A}'$ from an arbitrary knowledge base $\mathcal{K}$ that are linear in the size of $\mathcal{K}$ is easily achieved by a straightforward conservative extension using auxiliary names for intermediate concept descriptions and individuals (e.g., see defn. of *simple concepts* in [11, 13]). In fact, one can go further and also disallow value restrictions on left-hand-sides of inclusion dependencies in a normalized $\mathcal{T}'$, e.g., by replacing subsets $\{\forall f.\text{A} \sqsubseteq \text{B}\}$ of $\mathcal{T}'$ with the following (where A$'$ is a fresh primitive concept):

$$\{\text{A}' \sqsubseteq \forall f.\text{A}, \neg\text{A}' \sqsubseteq \forall f.\neg\text{A}, \text{A}' \sqsubseteq \text{B}\}.$$

Finally note that, when the UNA is assumed, it becomes necessary to distinguish the individuals that appear in the original ABox from those introduced during the normalization and for which, technically, the UNA does *not* apply. In particular, occurrences of path function assertions in a $\mathcal{CFD}_{nc}^{\forall,\neg}$ KB $\mathcal{K}$ can indeed influence the computational properties of reasoning problems for $\mathcal{K}$, see Section 4.

3 TBox Consistency & Concept Satisfiability

Unlike the case for $\mathcal{CFD}_{nc}$ and many other *lightweight logics*, it is possible for a $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox $\mathcal{T}$ to be inconsistent. In this section, we introduce a NLOGSPACE procedure for deciding TBox consistency for $\mathcal{CFD}_{nc}^{\forall,\neg}$.

Definition 3 Let PC and F be finite sets of primitive concepts and features, respectively. We define an *implication graph* over PC and F to be a node and edge labeled graph $G = (N, E)$ whose nodes are

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

and whose edges are labeled by $\mathsf{F} \cup \{\epsilon\}$ and such that $L \xrightarrow{\epsilon} L$, $\forall f.\mathrm{A} \xrightarrow{f} \mathrm{A}$, and $\neg\forall f.\mathrm{A} \xrightarrow{f} \neg\mathrm{A}$ are among the edges occurring in E. □

In the following, the nodes of implication graphs are identified with their labels. We call the pairs $(\mathrm{A}, \neg\mathrm{A})$ and $(\forall f.\mathrm{A}, \neg\forall f.\mathrm{A})$ complementary, and, if L is a node in an implication graph G, write $\neg L$ to denote the node of G for the concept label that is complementary to L. We call a node L in the implication graph $G = (N, E)$ an *empty node* if $L \xrightarrow{\epsilon} \neg L \in E$. We consider *paths* in G to be synonymous with words in a finite automaton based on G in which ϵ denotes the empty transition. Thus we identify, e.g., $\epsilon f \epsilon$ with ϵf with f, etc.

Definition 4 Let $G = (N, E)$ be an implication graph. We say that G is *closed under consequences* if $L_1 \xrightarrow{\epsilon} \neg L_2 \in E$ whenever for nodes L_1, L_2, and L there is a path $\mathsf{Pf} \subseteq E$ such that $L_1 \xrightarrow{\mathsf{Pf}} L$ and $L_2 \xrightarrow{\mathsf{Pf}} \neg L$. □

It is easy to see from the above that it is sufficient to consider paths of the form ϵ and $\epsilon f \epsilon$ to obtain the same result.

Definition 5 (Implication Graph for $\mathcal{T}$) Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox in normal form. We define an *implication graph for $\mathcal{T}$* to be the least graph $G = (N, E)$ over the primitive concepts and features occurring in $\mathcal{T}$ such that

- if $C \sqsubseteq D \in \mathcal{T}$ then $C \xrightarrow{\epsilon} D \in E$, and
- G is closed under consequences. □

The implication graph for $\mathcal{T}$ makes all implications between literals implied by $\mathcal{T}$ *explicit*: for $\mathcal{T} \models L_1 \sqsubseteq L_2$ it cannot be the case that $L_1 \xrightarrow{\epsilon} L_2 \notin E$ since $L_1 \sqcap \neg L_2$ must be empty, and this can only be enforced in the case where two paths traversing the same features and originating in L_1 and $\neg L_2$, respectively, end in a complementary pair of concepts. This yields the following theorem:

Theorem 6 Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox in normal form. Then $\mathcal{T}$ is consistent if and only if no two complementary nodes are empty in the implication graph G for $\mathcal{T}$.

Proof (sketch) If two complementary nodes, e.g., A and $\neg A$, are empty then $\mathcal{T} \models A \sqsubseteq \neg A$ and $\mathcal{T} \models \neg A \sqsubseteq A$; a contradiction.

If no such pair exists then there must be at least one non-empty node. We define an interpretation $\mathcal{I}$ for $\mathcal{T}$ to be a complete F-tree whose nodes form the domain $\triangle$ and whose edges provide the interpretation for features in F. We label the nodes of the tree by sets of literals as follows. First, initialize all labels to be empty, and then repeat the following: let n be a node that has *not* been

```
Data: A $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox $\mathcal{T}$
initialize $N$ and $E$ as in Definition 3;
$E := E \cup \{C \xrightarrow{\epsilon} D \mid C \sqsubseteq D \in \mathcal{T}\}$;
repeat
    for $L_1, L_2, L \in N$ and $\mathsf{Pf} \in \{\epsilon, \epsilon f \epsilon\}$ do
        if $L_1 \xrightarrow{\mathsf{Pf}} L, L_2 \xrightarrow{\mathsf{Pf}} \neg L \in E$ and $L_1 \xrightarrow{\epsilon} \neg L_2 \notin E$ then
            $E := E \cup \{L_1 \xrightarrow{\epsilon} \neg L_2\}$;
        end
    end
until $E$ is unchanged;
```

Algorithm 1: Construction of Implication Graph for $\mathcal{T}$

labeled by neither A nor $\neg A$ for some primitive concept A, but whose ancestors have already been labeled by all primitive concepts or their negations. Then either A or $\neg A$ must be non-empty (since $\mathcal{T}$ is consistent) and neither of the two literals is implied by any of the existing labels of n (since, otherwise, the label n would *already* contain such a literal). Assume A is non-empty. We add a literal L to the label of the node m if $\mathsf{Pf}(n) = m$ (i.e., m is a Pf-descendant of n) and there is a path Pf from A to L in G (note that the ϵ edges behave as empty transitions in the construction of the path; hence, e.g., the node n is labeled by A). The construction of G for $\mathcal{T}$ guarantees that no node will be labeled by complementary literals. Hence, since every node is labeled either by a primitive concept or by its negation and the labeling respects all constraints implied by $\mathcal{T}$, it can serve as the interpretation of primitive concepts, completing the construction. □

Observe that, since the above interpretation is a tree, all PFDs in $\mathcal{T}$ are satisfied vacuously and are therefore not taken into account in the above construction (nor in the definition of an implication graph). It is easy to see that Algorithm 1 constructs an implication graph for TBox $\mathcal{T}$ and runs in PTIME since there are only $|\mathcal{T}|^2$ edges that can be added to E. A straightforward but tedious modification of Algorithm 1 that (repeatedly) recomputes consequences as additional edges in the graph on the fly, instead of materializing all consequences as additional edges in the graph, can be shown to be NLOGSPACE-complete, completeness following from the need to explore paths in G.

Corollary 7 Consistency of $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBoxes is NLOGSPACE-complete.

Whenever $\mathcal{T}$ is consistent, the implication graph for $\mathcal{T}$ also records which primitive classes (and negations) are empty in every model, those for which $A \xrightarrow{\epsilon} \neg A \in E$. Thus

Corollary 8 Concept satisfiability with respect to $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBoxes is NLOGSPACE-complete.

4 ABox Reasoning and $\mathcal{K}$ Satisfiability

The complexity landscape for ABox reasoning, in particular for knowledge base consistency, depends crucially on the occurrence of non-key PFDs in $\mathcal{T}$, on path function assertions in $\mathcal{A}$ that are not equivalent to basic function assertions, and on whether UNA is assumed.

4.1 The Tractable Cases

We first consider KB consistency for primitive ABoxes and TBoxes without PFDs.

Definition 9 (2-CNF for an ABox) Let $\mathcal{T}$ be a consistent TBox without any mention of the PFD concept constructor and $\mathcal{A}$ a primitive ABox. We define 2-CNF$(\mathcal{T},\mathcal{A})$ to be the union of each of the following sets of clauses over the propositions $A(a)$ and $\forall f.A(a)$, where a is an ABox individual, A a primitive concept, and f a primitive feature, and where L_i stand for propositions or their negations.

- $\{\text{“}A(a)\text{”} : A(a) \in \mathcal{A}\}$
- $\{\text{“}L_1(a) \rightarrow L_2(a)\text{”} : \mathcal{T} \models L_1 \sqsubseteq L_2\}$
- $\{\text{“}L_1(a) \rightarrow L_2(b)\text{”} : f(a) = b \in \mathcal{A}, \mathcal{T} \models L_1 \sqsubseteq \forall f.L_2\}$
- $\{\text{“}L_1(b) \rightarrow L_2(a)\text{”} : f(a) = b \in \mathcal{A}, \mathcal{T} \models \forall f.L_1 \sqsubseteq L_2\}$

Recall that we have shown in Section 3 that any implication questions of the form $\mathcal{T} \models L_1 \sqsubseteq L_2$ can be *read* directly from the implication graph G for $\mathcal{T}$ since all implications of this form are explicit in G. Thus, we have the following.

Theorem 10 Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox without any mention of the PFD concept constructor and $\mathcal{A}$ a primitive ABox. Then the knowledge base $\mathcal{K} = (\mathcal{T},\mathcal{A})$ is consistent if and only if $\mathcal{T}$ is consistent and 2-CNF$(\mathcal{T},\mathcal{A})$ is satisfiable.

Proof (sketch) The satisfying assignment for 2-CNF$(\mathcal{T},\mathcal{A})$ yields an interpretation for the ABox of $\mathcal{K}$: $a^{\mathcal{I}} \in A^{\mathcal{I}}$ if the proposition $A(a)$ is true in the assignment. To complete this interpretation, since ABox individuals may be missing some features, we follow the construction from Theorem 6.

Conversely, if $\mathcal{K}$ is consistent, the class membership of ABox objects yields a satisfying assignment for 2-CNF$(\mathcal{T},\mathcal{A})$. □

Note that the theorem can be slightly strengthened by allowing general ABoxes since, without PFD constraints, there is never a need to equate ABox objects and thus reasoning under UNA is the same as reasoning without UNA.

Corollary 11 $\mathcal{CFD}_{nc}^{\forall,\neg}$ knowledge base consistency is NLOGSPACE-complete.

Adding Keys under UNA. We say that a PFD key constraint $L_1 \sqsubseteq L_2 : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$ is *potentially violated* by an ABox $\mathcal{A}$ with respect to two distinct individuals a and b if for all $0 < i \leq k$ the path functions $\mathsf{Pf}'_i(a)$ and $\mathsf{Pf}'_i(b)$ reach the same object in $\mathcal{A}$, where Pf'_i are the shortest prefixes of Pf_i with that property but where Pf'_i is not a prefix of Pf. Under UNA, such potentially violated PFDs can only be satisfied if the objects a and b do *not simultaneously* belong to descriptions L_1 and L_2, respectively, since UNA prevents us from *repairing* the violation by equating the objects $\mathsf{Pf}(a)$ and $\mathsf{Pf}(b)$. Consequently, one can simply add the following 2-CNF clauses,

- "$L_1(a) \to \neg L_2(b)$" and "$L_2(b) \to \neg L_1(a)$", for all pairs of individuals a, b in $\mathcal{A}$ and key PFDs $L_1 \sqsubseteq L_2 : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$ in $\mathcal{T}$ for which the latter is potentially violated by the former w.r.t. a and b,

called KEY-2-CNF($\mathcal{T}, \mathcal{A}$), to the set of clauses 2-CNF($\mathcal{T}, \mathcal{A}$) to account for this situation (recall Definition 9 above for the latter):

Theorem 12 Let $\mathcal{T}$ be a $\mathcal{CFD}_{nc}^{\forall,\neg}$ TBox with arbitrary occurrences of key PFDs and let $\mathcal{A}$ be a primitive ABox. Then, assuming UNA, the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is consistent if and only if $\mathcal{T}$ is consistent and 2-CNF($\mathcal{T}, \mathcal{A}$) $\cup$ KEY-2-CNF($\mathcal{T}, \mathcal{A}$) is satisfiable.

Proof (sketch) The proof is essentially the same as for Theorem 10 since the interpretation of the ABox makes all PFDs in $\mathcal{T}$ satisfied (the KEY-2-CNF($\mathcal{T}, \mathcal{A}$) clauses guarantee that) and since the additional anonymous objects cannot violate PFDs since corresponding parts of the interpretation are tree shaped. □

The theorem assumes that the ABox is primitive. However, this condition can be relaxed without harm to allow path assertions of the form "$f(a) = g(b)$" for which the same construction and proof argument would apply.

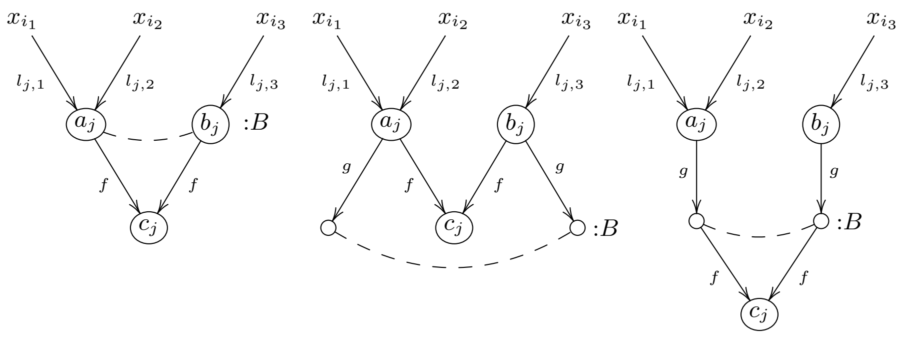
Corollary 13 $\mathcal{CFD}_{nc}^{\forall,\neg}$ knowledge base consistency is NLOGSPACE-complete for knowledge bases with primitive ABoxes, key-PFDs, and assuming unique names.

4.2 The Intractable Cases

Unfortunately, relaxing any of the above conditions leads to intractability. For each of the cases, we show a reduction of 3-SAT knowledge base consistency. Figure 2 illustrates the ABox *widgets* used to capture the behavior of 3-clauses in the three respective cases.

The Keys without UNA Case. We reduce 3-CNF satisfiability to KB consistency as follows: let φ be a propositional formula in 3-CNF with clauses $C_1, \ldots, C_k$ and propositional variables $x_1, \ldots, x_n$. We define a $\mathcal{CFD}_{nc}^{\forall,\neg}$ knowledge base $\mathcal{K}_\varphi = (\mathcal{T}, \mathcal{A})$ as follows:

1. For each propositional variable x_i in φ, introduce an $\mathcal{A}$ individual x_i.



The Key w/o UNA Case   The General FD Case   The Non-unit Paths Case

Fig. 2. 3-SAT Reduction Widgets for Clause C_j.

2. For each clause C_j introduce individuals a_j, b_j, and c_j.
3. For the variables x_{i_1}, x_{i_2}, and x_{i_2} that appear in a clause C_j, include the following basic function assertions in $\mathcal{A}$: $l_{j,1}(x_{i_1}) = a_j$, $l_{j,2}(x_{i_2}) = a_j$, and $l_{j,3}(x_{i_3}) = b_j$.
4. Add assertions $f(a_j) = c_j$ and $f(b_j) = c_j$ to $\mathcal{A}$.
5. Add concept assertion $B(b_j)$ to $\mathcal{A}$.
6. Add the following inclusion dependencies to $\mathcal{T}$:
   - $T \sqsubseteq \forall l_{j,1}.\neg A$ and $\neg T \sqsubseteq \forall l_{j,1}.A$ when x_{i_1} appears positively in C_j, and $T \sqsubseteq \forall l_{j,1}.A$ and $\neg T \sqsubseteq \forall l_{j,1}.\neg A$ when x_{i_1} appears negatively in C_j;
   - $T \sqsubseteq \forall l_{j,2}.B$ and $\neg T \sqsubseteq \forall l_{j,2}.\neg B$ when x_{i_2} appears positively in C_j, and $T \sqsubseteq \forall l_{j,2}.\neg B$ and $\neg T \sqsubseteq \forall l_{j,2}.B$ when x_{i_2} appears negatively in C_j; and
   - $T \sqsubseteq \forall l_{j,3}.\neg A$ and $\neg T \sqsubseteq \forall l_{j,3}.A$ when x_{i_3} appears positively in C_j, and $T \sqsubseteq \forall l_{j,3}.A$ and $\neg T \sqsubseteq \forall l_{j,3}.\neg A$ when x_{i_3} appears negatively in C_j.
7. Finally, add the key PFD $A \sqsubseteq A : f \to \mathit{id}$ to $\mathcal{T}$.

A truth assignment to the propositions x_i occurring in φ is then encoded by an interpretation $\mathcal{I}$ of $\mathcal{K}_\varphi$: $x_i^\mathcal{I} \in T^\mathcal{I}$ is where x_i is true, and $x_i^\mathcal{I} \in \neg T^\mathcal{I}$ is where x_i is false. With this in mind, it is straightforward (albeit tedious) to confirm that any interpretation of $\mathcal{K}_\varphi$ will only encode a *satisfying* truth assignment and that, in turn, an interpretation of $\mathcal{K}_\varphi$ can always be constructed by a satisfying truth assignment.

The General Functional Dependencies Case. Allowing *functional dependencies* (i.e., non-key PFDs) even under UNA also leads to intractability. The reduction is similar to the one above; the first four steps are identical:

5. Add concept assertion $\forall g.B(b_j)$ to $\mathcal{A}$.
6. Add the following to $\mathcal{T}$:
   - $T \sqsubseteq \forall l_{j,1}.\neg A$ and $\neg T \sqsubseteq \forall l_{j,1}.A$ when x_{i_1} appears positively in C_j, and $T \sqsubseteq \forall l_{j,1}.A$ and $\neg T \sqsubseteq \forall l_{j,1}.\neg A$ when x_{i_1} appears negatively in C_j.

  - $T \sqsubseteq \forall l_{j,2}.\forall g.B$ and $\neg T \sqsubseteq \forall l_{j,2}.\forall g.\neg B$ when x_{i_2} appears positively in C_j, and $T \sqsubseteq \forall l_{j,2}.\forall g.\neg B$ and $\neg T \sqsubseteq \forall l_{j,2}.\forall g.B$ when x_{i_2} appears negatively in C_j; and
  - $T \sqsubseteq \forall l_{j,3}.\neg A$ and $\neg T \sqsubseteq \forall l_{j,3}.A$ when x_{i_3} appears positively in C_j, and $T \sqsubseteq \forall l_{j,3}.A$ and $\neg T \sqsubseteq \forall l_{j,3}.\neg A$ when x_{i_3} appears negatively in C_j.

7. $A \sqsubseteq A : f \rightarrow g \in \mathcal{T}$.

The Non-unit Path Agreements Case. Similarly, allowing non-primitive ABoxes with path function assertions even under UNA leads to intractability. The reduction is again similar; the first three steps are identical to the previous cases:

4. Add $g.f(a_j) = c_j$, $g.f(b_j) = c_j$.
5. Add $\forall g.B(b_j)$ to $\mathcal{A}$.
6. Add the following to $\mathcal{T}$:
   - $T \sqsubseteq \forall l_{j,1}.\forall g.\neg A$ and $\neg T \sqsubseteq \forall l_{j,1}.\forall g.A$ when x_{i_1} appears positively in C_j, and $T \sqsubseteq \forall l_{j,1}.\forall g.A$ and $\neg T \sqsubseteq \forall l_{j,1}.\forall g.\neg A$ when x_{i_1} appears negatively in C_j.
   - $T \sqsubseteq \forall l_{j,2}.\forall g.B$ and $\neg T \sqsubseteq \forall l_{j,2}.\forall g.\neg B$ when x_{i_2} appears positively in C_j, and $T \sqsubseteq \forall l_{j,2}.\forall g.\neg B$ and $\neg T \sqsubseteq \forall l_{j,2}.\forall g.B$ when x_{i_2} appears negatively in C_j; and
   - $T \sqsubseteq \forall l_{j,3}.\forall g.\neg A$ and $\neg T \sqsubseteq \forall l_{j,3}.\forall g.A$ when x_{i_3} appears positively in C_j, and $T \sqsubseteq \forall l_{j,3}.\forall g.A$ and $\neg T \sqsubseteq \forall l_{j,3}.\forall g.\neg A$ when x_{i_3} appears negatively in C_j.
7. Finally, add the key PFD $A \sqsubseteq A : f \rightarrow id$ to $\mathcal{T}$.

In all three cases it is easy to confirm that:

Lemma 14 Let φ be a propositional formula in 3-CNF. Then φ is satisfiable if and only if $\mathcal{K}_\varphi$ is consistent.

Proof (sketch) It is easy to verify that a satisfying assignment for φ can be used to create a model for $\mathcal{K}$, essentially by assigning the class membership of the individuals x_{i_j} to the primitive concepts T and F and then extending this assignment to the a_j, b_j and c_j individuals as dictated by the constraints in $\mathcal{T}$.

On the other hand, a model for $\mathcal{K}$ (when restricted to each widget representing a clause) guarantees that at least one of the x_{i_1}, x_{i_2}, and x_{i_3} will belong to the concept T: the concept B is used to make the widget corresponding to a 3-CNF clause unsatisfied exactly when the clause itself is unsatisfied: an interpretation assigning all of x_{i_1}, x_{i_2}, and x_{i_3} to F forces the two individuals connected by the dashed lines in the widgets in Figure 2 to be equal as a consequence of the PFD and to belong to B and $\neg B$ at the same time, thus disqualifying such interpretations as models of $\mathcal{K}$.

Note the use of paths of length 2 to construct anonymous objects to evade UNA. Hence, knowledge base consistency without assuming UNA or when allowing functional dependencies in $\mathcal{T}$ or non-primitive path equalities in $\mathcal{A}$ is

NP-complete (membership in NP is established by guessing primitive classes and equalities for ABox individuals).

Instance Retrieval. Since $\mathcal{CFD}_{nc}^{\forall,\neg}$ is closed under negation, *instance retrieval*, the question $\mathcal{K} \models C(a)$ reduces naturally to knowledge base (in)consistency, i.e., $\mathcal{K} \models C(a)$ if and only if $(\mathcal{T}, \mathcal{A} \cup \{\neg C(a)\})$ is inconsistent (for $\mathcal{K} == (\mathcal{T}, \mathcal{A})$) and therefore inherits the computational properties of testing for consistency:

- $\mathcal{CFD}_{nc}^{\forall,\neg}$ instance retrieval is in NLOGSPACE-complete for primitive ABoxes and key PFDs under UNA, and
- coNP-complete in all other cases.

The instance retrieval result cannot, however, be extended to conjunctive queries. In particular, it is a straightforward consequence of results of Schaerf that allowing negated primitive concepts on left-hand-sides of inclusion dependencies in $\mathcal{CFD}_{nc}$ will alone make CQ answering coNP-complete [10].

5 Related Work and Summary Comments

PFD-based constraints were first introduced and studied in the context of graph-oriented data models (similar to RDF) and its refinements [7, 16]. Subsequently, an FD concept constructor was proposed and incorporated in Classic [3], an early DL with PTIME reasoning capabilities, without changing the complexity of its implication problem. On the other hand, removing the restrictions imposed on PFDs (see Section 2 (1)), makes logical implication EXPTIME-complete [9] and general reasoning, in particular knowledge base consistency, undecidable [13].

Logics in the $\mathcal{CFD}$ family that allow conjunctions on left of subsumptions [14] cannot support tractability in the presence of, e.g., disjointness. The most notable cases are as follows.

- Allowing conjunction "$\sqcap$" yields the logic $\mathcal{CFD}^{\perp}$ and therefore makes reasoning PSPACE-complete [14].
- Allowing conjunction and value restriction "$\forall$" makes reasoning EXPTIME-complete even in the absence of negation (or the empty concept $\perp$) [9].

Adding various forms of functional dependencies and keys to other DLs—usually as additional separate varieties of constraints, often called a *key box*—have been considered, e.g., by [6]. They show that their dialect is undecidable for description logics with inverse roles, but becomes decidable when unary functional dependencies are disallowed. This line of investigation is continued in the context of PFDs and inverse features, with analogous results [12]. Subsequently, Calvanese et al. have shown how DL-Lite can be extended with a path-based variety of identification constraints analogous to PFDs without affecting the complexity of reasoning problems [5].

In this paper, we have considered extensions to the logic $\mathcal{CFD}_{nc}$ and their impact on the complexity of reasoning problems. $\mathcal{CFD}_{nc}^{\forall,\neg}$ and its various restricted fragments introduced in this paper have served as vehicles for this study. Most significantly, we have shown that instance retrieval remains in PTIME for $\mathcal{CFD}_{nc}^{\forall,\neg}$ under UNA, provided that ABoxes are primitive and that PFDs occurring in TBoxes are keys. Thus, *basic graph pattern* (BGP) evaluation over $\mathcal{CFD}_{nc}^{\forall,\neg}$ knowledge bases satisfying these conditions is also in PTIME.

References

1. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *J. Artificial Intelligence Research*, 36:1–69, 2009.
2. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.
3. Alexander Borgida and Grant Weddell. Adding Uniqueness Constraints to Description Logics (Preliminary Report). In *International Conference on Deductive and Object-Oriented Databases*, pages 85–102, 1997.
4. 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. of Automated Reasoning*, 39(3):385–429, 2007.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-Based Identification Constraints in Description Logics. In *Principles of Knowledge Representation and Reasoning*, pages 231–241, 2008.
6. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 155–160, 2001.
7. Minoru Ito and Grant Weddell. Implication Problems for Functional Constraints on Databases Supporting Complex Objects. *Journal of Computer and System Sciences*, 49(3):726–768, 1994.
8. 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.
9. Vitaliy L. Khizder, David Toman, and Grant Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *Int. Conf. on Database Theory ICDT'01*, pages 54–67, 2001.
10. Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *J. Intell. Inf. Syst.*, 2(3):265–278, 1993.
11. David Toman and Grant Weddell. On Keys and Functional Dependencies as First-Class Citizens in Description Logics. In *Proc. of Int. Joint Conf. on Automated Reasoning (IJCAR)*, pages 647–661, 2006.
12. David Toman and Grant E. Weddell. On the interaction between inverse features and path-functional dependencies in description logics. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 603–608, 2005.
13. David Toman and Grant E. Weddell. On keys and functional dependencies as first-class citizens in description logics. *J. Autom. Reasoning*, 40(2-3):117–132, 2008.
14. 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.
15. 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.
16. Grant Weddell. A Theory of Functional Dependencies for Object Oriented Data Models. In *International Conference on Deductive and Object-Oriented Databases*, pages 165–184, 1989.