On Path-functional Dependencies as First-class Citizens in Description Logics

David Toman and Grant Weddell

School of Computer Science, University of Waterloo
Email: {david,gweddell}@uwaterloo.ca

Abstract

We investigate whether *path-functional dependencies* can be granted full status as a concept constructor in a Boolean-complete description logic. In particular, we show that this leads to undecidability of the associated logical implication problem if such dependencies are allowed within the scope of a negation or on the left-hand-side of inclusion dependencies. We then show that allowing such dependencies to occur in the scope of *monotone* concept constructors on the right-hand-side of inclusion dependencies will still lead to decidable implication problems.

1 Introduction

To date, description logics (DLs) have incorporated keys or functional dependencies in one of two ways. The first introduces a separate family of constraints, e.g., in the form of a *key box* [5, 6, 12, 13], while the second incorporates such constraints by introducing a concept constructor called a *path-functional dependency* (PFD) [10, 16, 17]. However, even the latter approach has so far fallen short of a "full integration" of keys or functional dependencies since occurrences of this constructor are essentially limited to top-level concept descriptions on right hand sides of inclusion dependencies. In this paper, we investigate whether such syntactic restrictions are necessary—unfortunately we show that indeed this is the case—and study the limits of decidability in such a setting. Our main contributions are as follows.

- We show in general that allowing PFDs in the scope of negation or on the left hand side of inclusion dependencies leads to undecidability.
- Conversely, we show that allowing PFDs in the scope of *monotone* concept constructors on the right hand side of inclusion dependencies leads to decidable implication problems.

1.1 Background and Related Work

PFDs were introduced and studied in the context of object-oriented data models in [8, 20]. In [4], an FD concept constructor was proposed and incorporated in Classic, an early DL with a PTIME reasoning procedure, without changing the complexity of its implication problem. In [10], it is shown that the generalization of this constructor to PFDs will alone lead to EXPTIME completeness of the implication problem; [16, 17] show that this complexity remains unchanged in the presence of very rich, e.g., Boolean complete DLs. Note that this earlier work assumes the above syntactic restrictions on occurrences of the PFD concept constructor in inclusion dependencies. PFDs have also been used in a number of applications in object-oriented schema diagnosis and synthesis [2, 3], in query optimization [7, 9] and in the selection of indexing for a database [14].

The remainder of the paper is organized as follows. The definition of $\mathcal{DLFD}$, a Boolean complete DL based on features that includes the PFD concept constructor is given next. In Section 3, we show that the interaction of this constructor with negation leads to undecidability. Section 4 then shows how decidability can be regained while still allowing PFDs in the scope of monotone concept constructors on the right hand sides of inclusion dependencies, most significantly in the scope of concept union and feature restriction. Our summary comments follow in Section 5.

2 Definitions

We use the following DL dialect in this paper:

Definition 1 (Description Logic $\mathcal{DLFD}$) *Let* F *and* C *be sets of attribute names and primitive concept names, respectively. A* path expression *is defined by the grammar "*$\mathsf{Pf} ::= f.\,\mathsf{Pf} \mid Id$*" for* $f \in \mathsf{F}$*. We define derived* concept descriptions *by the grammar on the left-hand-side of Figure 1. A concept description obtained by using the final production of this grammar is called a* path functional dependency *(PFD).*

An inclusion dependency $\mathcal{C}$ *is an expression of the form* $D \sqsubseteq E$*. A* terminology $\mathcal{T}$ *consists of a finite set of inclusion dependencies.*

The semantics *of expressions is defined with respect to a structure* $(\Delta, \cdot^{\mathcal{I}})$*, where* Δ *is a domain of "objects" and* $(.)^{\mathcal{I}}$ *an interpretation function that fixes the interpretation of primitive concepts* C *to be subsets of* Δ *and primitive attributes* f *to be total functions* $(f)^{\mathcal{I}} : \Delta \to \Delta$*. The interpretation is extended to path expressions,* $(Id)^{\mathcal{I}} = \lambda x.x$*,* $(f.\,\mathsf{Pf})^{\mathcal{I}} = (\mathsf{Pf})^{\mathcal{I}} \circ (f)^{\mathcal{I}}$ *and derived concept descriptions* D *and* E *as defined on the right-hand-side of Figure 1.*

An interpretation satisfies an inclusion dependency $D \sqsubseteq E$ *if* $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$*. The* logical implication problem *asks if* $\mathcal{T} \models D \sqsubseteq E$ *holds; that is, for a* posed

Syntax		Semantics: Defn of "$(\cdot)^{\mathcal{I}}$"
$D, E ::=$	C	$(C)^{\mathcal{I}} \subseteq \Delta$
	$D_1 \sqcap D_2$	$(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$
|	$D_1 \sqcup D_2$	$(D_1)^{\mathcal{I}} \cup (D_2)^{\mathcal{I}}$
|	$\forall f.D$	$\{x : (f)^{\mathcal{I}}(x) \in (D)^{\mathcal{I}}\}$
|	$\neg D$	$\Delta \setminus (D)^{\mathcal{I}}$
|	$D : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}$	$\{x : \forall\, y \in (D)^{\mathcal{I}}.$ $\bigwedge_{i=1}^{k} (\mathsf{Pf}_i)^{\mathcal{I}}(x) = (\mathsf{Pf}_i)^{\mathcal{I}}(y) \Rightarrow (\mathsf{Pf})^{\mathcal{I}}(x) = (\mathsf{Pf})^{\mathcal{I}}(y)\}$

Figure 1: Syntax and Semantics of $\mathcal{DLFD}$.

question $D \sqsubseteq E$, *if* $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$ *for all interpretations that satisfy all inclusion dependencies in* $\mathcal{T}$.

3 Undecidability

First we show that allowing an arbitrary use of the PFD concept constructor leads to an undecidable implication problem. We show a reduction of the unrestricted tiling problem to the $\mathcal{DLFD}$ implication problem. An instance U of the unrestricted tiling problem is a triple (T, H, V) where T is a finite set of tile types and $H, V \subseteq T \times T$ two binary relations. A *solution* to T is a mapping $t : \mathbf{N} \times \mathbf{N} \rightarrow T$ such that $(t(i,j), t(i+1,j)) \in H$ and $(t(i,j), t(i,j+1)) \in V$ for all $i \in \mathbf{N}$. This problem is Π_0^0-complete [1, 19]. The main part of the reduction constructs a terminology for a given tiling problem, denoted $\mathcal{T}(T, H, V)$, in the following four steps.

1. To establish an *integer grid*, begin by enforcing the creation of an infinite *horizontal sequence* of objects that are instances of alternating concepts C and D.

$$\begin{aligned} \mathrm{C} &\sqsubseteq (\forall f.\mathrm{C}_f) \sqcap (C : f \rightarrow Id) \sqcap (\forall g.\mathrm{C}_g) \sqcap (C : g \rightarrow Id) \\ \mathrm{D} &\sqsubseteq (\forall f.\mathrm{D}_f) \sqcap (D : f \rightarrow Id) \sqcap (\forall g.\mathrm{D}_g) \sqcap (D : g \rightarrow Id) \\ (\mathrm{C}_f \sqcup \mathrm{D}_f) &\sqsubseteq \neg(\mathrm{C}_g \sqcup \mathrm{D}_g) \\ \mathrm{C} &\sqsubseteq \neg(D : g \rightarrow f) \\ \mathrm{D} &\sqsubseteq \neg(C : f \rightarrow g) \end{aligned}$$

   The infinite sequence is enforced by requiring the existence of a single instance of $\mathrm{C} \sqcap \forall f.\neg \mathrm{D}_f$ (see later); this is illustrated by the first two rows of concepts at the top of Figure 2.

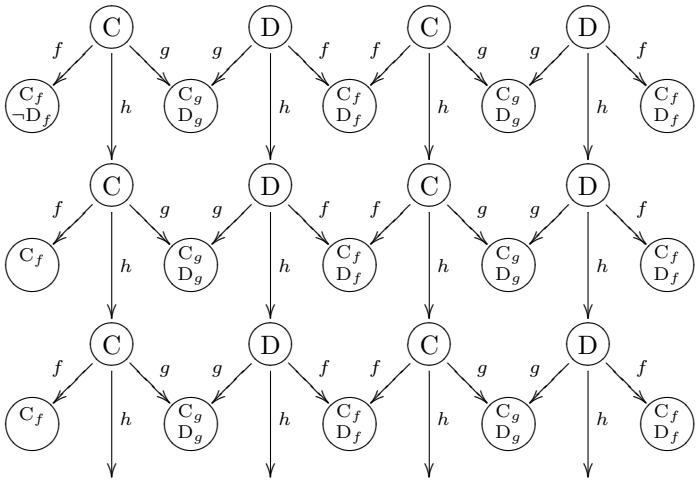

Figure 2: Defining a Grid.

2. Then extend this sequence in the vertical direction to form squares.

$$\begin{array}{rcl} \mathrm{C} & \sqsubseteq & (\forall h.C) \sqcap (\mathrm{D} : g \rightarrow h.g) \\ \mathrm{D} & \sqsubseteq & (\forall h.D) \sqcap (\mathrm{C} : f \rightarrow h.f) \\ (\mathrm{C} \sqcup \mathrm{D}) & \sqsubseteq & (\mathrm{C} \sqcup \mathrm{D}) : h \rightarrow Id \end{array}$$

   The effect of these additional assertions is depicted in the lower part of Figure 2.

3. Introduce primitive concepts T_i for each tile type in T, and ensure the types are distinct.

$$\begin{array}{rcl} \mathrm{C} \sqcup \mathrm{D} & \sqsubseteq & (\bigsqcup \mathrm{T}_i) \\ \mathrm{T}_i \sqcap \mathrm{T}_j & \sqsubseteq & \bot \quad \text{for } i < j \end{array}$$

4. And finally, enforce the tiling constraints.

$$\begin{array}{lrcl} (\textit{vertical adjacency}) & \mathrm{T}_i \sqcap \forall h.\mathrm{T_j} & \sqsubseteq & \bot \quad \text{for } (i,j) \notin V \\ (\textit{horizontal adjacency}) & \mathrm{T}_i \sqcap \mathrm{C} & \sqsubseteq & (\mathrm{T}_j \sqcap \mathrm{D}) : g \rightarrow Id \quad \text{and} \\ & \mathrm{T}_i \sqcap \mathrm{D} & \sqsubseteq & (\mathrm{T}_j \sqcap \mathrm{C}) : f \rightarrow Id \quad \text{for } (i,j) \notin H \end{array}$$

The construction above yields immediately the following result:

Theorem 2 *An instance (T, H, V) of the infinite tiling problem admits a solution if and only if*

$$\mathcal{T}(T, H, V) \not\models \mathrm{C} \sqcap \forall f.\neg \mathrm{D}_f \sqsubseteq \bot.$$

Corollary 3 *The logical implication problem for $\mathcal{DLFD}$ is undecidable.*

4 On Regaining Decidability

We now show that undecidability is a consequence of allowing PFDs to occur within the scope of negation. In particular, and for the remainder of the paper, we shall assume a *limited* $\mathcal{DLFD}$ in which inclusion dependencies, $\mathrm{D} \sqsubseteq \mathrm{E}$, are presumed to adhere to the following less general grammar.

$$\begin{array}{rcl} D & ::= & C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \mid \neg D \\ E & ::= & D \mid E_1 \sqcap E_2 \mid E_1 \sqcup E_2 \mid \forall f.E \mid D : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf} \end{array}$$

Observe that PFDs must now occur on right hand sides of inclusion dependencies at either the top level or *within the scope of monotone concept constructors*; this implies that limited $\mathcal{DLFD}$ is a strict generalization of earlier dialects. Note that allowing PFDs on left hand sides is equivalent to allowing PFDs in the scope of negation:

Example 4 $\mathrm{C} \sqsubseteq \neg(D : f \rightarrow g)$ is equivalent to $\mathrm{C} \sqcap (D : f \rightarrow g) \sqsubseteq \bot$.

In the following, we reduce logical implication problems in limited $\mathcal{DLFD}$ to simpler formulations for which existing decisions procedures can be applied [16].

4.1 Transformation of Terminologies

We start by showing that allowing PFDs in *monotone* concept constructors within terminologies can be avoided by a syntactic transformation.

Definition 5 (Simple Constraints and Terminologies) *An inclusion dependency* $\mathrm{D} \sqsubseteq \mathrm{E} \in \mathcal{T}$ *is called* simple *if it conforms to limited* $\mathcal{DLFD}$ *and if the right hand side can be parsed by the following grammar.*

$$E ::= D \mid D : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}$$

A terminology $\mathcal{T}$ *is called* simple *if all its inclusion dependencies are simple.*

For a given terminology $\mathcal{T}$, we construct a simple terminology $\mathcal{T}^{\mathrm{simp}}$ by rewriting the right hand sides of inclusion dependencies as follows:

$$\begin{array}{rcl} (\mathrm{D} \sqsubseteq \mathrm{D}')^{\mathrm{simp}} & = & \{\mathrm{D} \sqsubseteq \mathrm{D}'\} \\ (\mathrm{D} \sqsubseteq \mathrm{E}_1 \sqcap \mathrm{E}_2)^{\mathrm{simp}} & = & \{\mathrm{D} \sqsubseteq \mathrm{D}_1 \sqcap \mathrm{D}_2\} \cup (\mathrm{D}_1 \sqsubseteq \mathrm{E}_1)^{\mathrm{simp}} \cup (\mathrm{D}_2 \sqsubseteq \mathrm{E}_2)^{\mathrm{simp}} \\ (\mathrm{D} \sqsubseteq \mathrm{E}_1 \sqcup \mathrm{E}_2)^{\mathrm{simp}} & = & \{\mathrm{D} \sqsubseteq \mathrm{D}_1 \sqcup \mathrm{D}_2\} \cup (\mathrm{D}_1 \sqsubseteq \mathrm{E}_1)^{\mathrm{simp}} \cup (\mathrm{D}_2 \sqsubseteq \mathrm{E}_2)^{\mathrm{simp}} \\ (\mathrm{D} \sqsubseteq \forall f.\mathrm{E}_1)^{\mathrm{simp}} & = & \{\mathrm{D} \sqsubseteq \forall f.\mathrm{D}_1\} \cup (\mathrm{D}_1 \sqsubseteq \mathrm{E}_1)^{\mathrm{simp}} \end{array}$$

for $\mathrm{D} \sqsubseteq \mathrm{D}'$ a simple inclusion dependency and D_1 and D_2 fresh primitive concept names. We define $\mathcal{T}^{\mathrm{simp}} = \bigcup_{\mathrm{D} \sqsubseteq \mathrm{E} \in \mathcal{T}} (\mathrm{D} \sqsubseteq \mathrm{E})^{\mathrm{simp}}$.

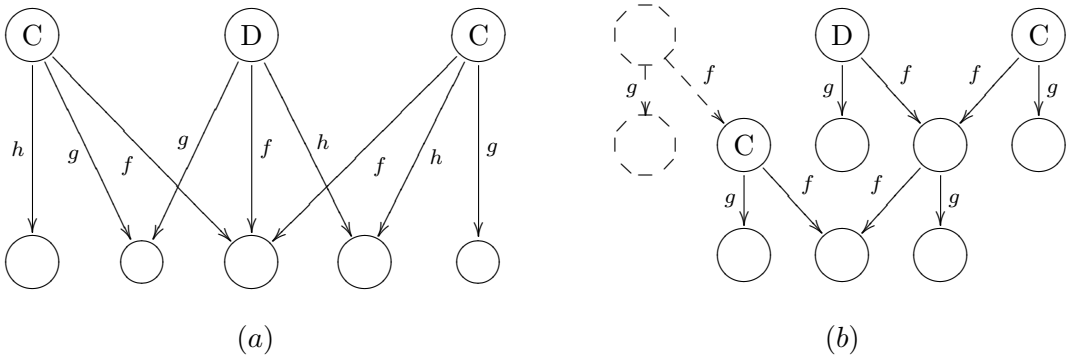

Figure 3: Counterexamples for Examples 8 and 9.

Lemma 6 *1. Let $\mathcal{I} \models \mathcal{T}^{\text{simp}}$. Then $\mathcal{I} \models \mathcal{T}$;*

2. Let $\mathcal{I} \models \mathcal{T}$. Then there is $\mathcal{I}'$ such that $\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of all symbols in $\mathcal{T}$ and $\mathcal{I}' \models \mathcal{T}^{\text{simp}}$.

Proof: Follows by straightforward inductions on the definition of $(\cdot)^{\text{simp}}$. □

Thus, in terminologies, the interaction of positive concept constructors with PFDs poses little difficulty and we can use existing decision procedures for the implication problem.

Theorem 7 *Let $\mathcal{T}$ be a terminology conforming to limited $\mathcal{DLFD}$ and $\mathcal{C}$ a simple inclusion dependency. Then $\mathcal{T} \models \mathcal{C}$ is decidable and complete for EXPTIME.*

Proof: The theorem is a consequence of Lemma 6 and of reductions presented in [16]. □

4.2 Transformation of Posed Questions

Now assuming, w.l.o.g., that a given terminology is simple, we exhibit a reduction of a logical implication problem with a posed question expressed in limited $\mathcal{DLFD}$. Unfortunately, allowing other than simple inclusion dependencies as posed questions leads to more complications. Consider the following two examples.

Example 8 A counterexample to $\text{D} \sqsubseteq (\text{C} : f, g \rightarrow h) \sqcup (\text{C} : f, h \rightarrow g)$ is depicted in Figure 3(a). Note that any such counterexample must also falsify

$\mathrm{C} \sqsubseteq \mathrm{C} : f \rightarrow Id$ since distinct C objects that agree on f will be required. Thus:

$$\{\mathrm{C} \sqsubseteq \mathrm{C} : f \rightarrow Id\} \models \mathrm{D} \sqsubseteq (\mathrm{C} : f, g \rightarrow h) \sqcup (\mathrm{C} : f, h \rightarrow g).$$

Example 9 A counterexample to $\mathrm{D} \sqsubseteq (\mathrm{C} : f \rightarrow g) \sqcup \forall f.(\mathrm{C} : f \rightarrow g)$ is shown in Figure 3(b). Observe with this case that distinct C objects must occur at *different* levels when compared to a D-rooted tree.

The examples illustrate a need for multiple *root objects* in counterexample interpretations, with the roots themselves occurring at different levels. Our overall strategy is to therefore reduce a logical implication problem to a negation of a consistency problem in an alternative formulation in which objects in a satisfying counterexample *denote up to ℓ possible copies in a counterexample interpretation for the original problem*, where ℓ is the number of occurrences of PFDs in the posed question.

To encode this one-to-many mapping of objects, we require a general way to have ℓ copies of concepts occurring in a given membership problem. We therefore write D^i to denote the concept description D in which all primitive concepts C are replaced by C^i. For a simple terminology $\mathcal{T}$ we then define

$$\begin{aligned}
\mathcal{T}^i &= \{\mathsf{Nd}^i \sqcap \mathrm{D}^i \sqsubseteq \mathrm{E}^i \mid \mathrm{D} \sqsubseteq \mathrm{E} \in \mathcal{T}, E \text{ a non PFD}\}, \text{ and} \\
\mathcal{T}^{i,j} &= \{\mathsf{Nd}^i \sqcap \mathsf{Nd}^j \sqcap \mathrm{D}^i \sqcap \mathrm{E}^j \sqcap (\sqcap_{1 \leq n \leq k} \forall\, \mathsf{Pf}_n\, .\mathsf{Eq}^{i,j}) \sqsubseteq \forall\, \mathsf{Pf}\, .\mathsf{Eq}^{i,j}, \\
&\qquad \mathsf{Nd}^i \sqcap \mathsf{Nd}^j \sqcap \mathrm{D}^j \sqcap \mathrm{E}^i \sqcap (\sqcap_{1 \leq n \leq k} \forall\, \mathsf{Pf}_n\, .\mathsf{Eq}^{i,j}) \sqsubseteq \forall\, \mathsf{Pf}\, .\mathsf{Eq}^{i,j} \\
&\qquad\qquad \mid \mathrm{D} \sqsubseteq \mathrm{E} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \in \mathcal{T}\ \}.
\end{aligned}$$

For a concept description E we define

$$\mathrm{Not}(\mathrm{E}) = \begin{cases}
\neg \mathrm{D}^0 & \text{if E } (= \mathrm{D}) \text{ is free of PFDs,} \\
\mathrm{Not}(\mathrm{E}_1) \sqcap \mathrm{Not}(\mathrm{E}_2) & \text{if } \mathrm{E} = \mathrm{E}_1 \sqcup \mathrm{E}_2, \\
\mathrm{Not}(\mathrm{E}_1) \sqcup \mathrm{Not}(\mathrm{E}_2) & \text{if } \mathrm{E} = \mathrm{E}_1 \sqcap \mathrm{E}_2, \\
\forall f.\mathrm{Not}(\mathrm{E}_1) & \text{if } \mathrm{E} = \forall f.\mathrm{E}_1, \\
\mathsf{Nd}^i \sqcap \mathrm{D}^i \sqcap (\sqcap_{1 \leq i \leq k} \forall\, \mathsf{Pf}_i\, .\mathsf{Eq}^{0,i}) \sqcap \forall\, \mathsf{Pf}\, .\neg \mathsf{Eq}^{0,i} & \\
\qquad\qquad \text{otherwise, when } \mathrm{E} = \mathrm{D} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.
\end{cases}$$

where i in the last equation is the index of the PFD in the original posed question.

In the above, we have introduced primitive concepts $\mathsf{Eq}^{i,j}, 0 \leq i \neq j \leq \ell$, to express that the ith and jth object copies coincide, and $\mathsf{Nd}^i, 0 \leq i \leq \ell$, to assert that the ith copy exists. The following auxiliary sets of constraints are therefore defined to account for the axioms of equality and for the fact that features in $\mathcal{DLFD}$ denote total functions.

$$\begin{aligned}
\mathcal{E}(l) &= \{\mathsf{Eq}^{i,j} \sqcap \mathsf{Eq}^{j,k} \sqsubseteq \mathsf{Eq}^{i,k} \mid 0 \leq i < j < k \leq l\} \\
&\cup\ \{\mathsf{Eq}^{i,j} \sqsubseteq \mathsf{Eq}^{j,i} \mid 0 \leq i < j \leq l\} \\
&\cup\ \{(\mathsf{Eq}^{i,j} \sqcap \mathrm{C}^i) \sqsubseteq \mathrm{C}^j \mid 0 \leq i \neq j \leq l \text{ and C a primitive concept}\} \\
&\cup\ \{\mathsf{Eq}^{i,j} \sqsubseteq \forall f.\mathsf{Eq}^{i,j} \mid 0 \leq i \neq j \leq l \text{ and } f \text{ a primitive feature}\} \\
\mathcal{N}(l) &= \{\mathsf{Nd}^i \sqsubseteq \forall f.\mathsf{Nd}^i \mid 0 \leq i \leq l \text{ and } f \text{ a primitive feature}\}
\end{aligned}$$

Theorem 10 *Let $\mathcal{T}$ be a simple terminology and* $\mathrm{D} \sqsubseteq \mathrm{E}$ *an inclusion dependency containing l occurrences of the PFD concept constructor. Then* $\mathcal{T} \models \mathrm{D} \sqsubseteq \mathrm{E}$ *if and only if*

$$(\bigcup_{0 \le i \le l} \mathcal{T}^i) \cup (\bigcup_{0 \le i < j \le l} \mathcal{T}^{i,j}) \cup \mathcal{E}(l) \cup \mathcal{N}(l) \models (\mathsf{Nd}^0 \sqcap \mathrm{D}^0 \sqcap \mathrm{Not(E)}) \sqsubseteq \bot.$$

Proof: (sketch) Given an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \not\models \mathrm{D} \sqsubseteq \mathrm{E}$ we construct an interpretation $\mathcal{J}$ as follows. First, in the construction, we use a many-to-one map $\delta : \Delta \to \Delta^{\mathcal{J}}$ to associate objects in $\mathcal{I}$ with thise in $\mathcal{J}$. The range of δ serves as the domain of the interpretation $\mathcal{J}$. For the counterexample object $o \in (\mathrm{D} \sqcap \neg \mathrm{E})^{\mathcal{I}}$ we set $\delta o \in (\mathsf{Nd}^0)^{\mathcal{J}}$. Then, for all $o \in \Delta$ and $0 \le i \ne j \le l$ we define the map δ and the interpretation $\mathcal{I}$ as follows:

- $\delta o \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge (f)^{\mathcal{I}}(o) = o' \Rightarrow \delta o' \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge (f)^{\mathcal{J}}(\delta o) = \delta o'$,
- $\delta o \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge o \in (D)^{\mathcal{I}} \Rightarrow \delta o \in (D)^{\mathcal{J}}$ for D a PFD-free concept,
- $\delta o = \delta o' \wedge \delta o \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge \delta o' \in (\mathsf{Nd}^j)^{\mathcal{J}} \wedge (\mathsf{Pf})^{\mathcal{I}}(o) = (\mathsf{Pf})^{\mathcal{I}}(o') \Rightarrow$ $\delta o \in (\mathsf{Eq}^{i,j})^{\mathcal{J}}$, and
- $\delta o \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge o \in (\neg \mathrm{D} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf})^{\mathcal{I}}$ where $\mathrm{D} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$ is the i-th PFD constructor in E. Thus there must be $o' \in \Delta$ such that $o' \in (\mathrm{D})^{\mathcal{I}}$ and the pair o, o' agrees on all Pf_i but disagrees on Pf; we set $\delta o = \delta o'$ and $\delta o' \in (\mathsf{Nd}^i \sqcap \mathrm{D}^i \sqcap (\sqcap_{1 \le i \le k} \forall\, \mathsf{Pf}_i\, .\mathsf{Eq}^{0,i}) \sqcap \forall\, \mathsf{Pf}\, .\neg \mathsf{Eq}^{0,i})^{\mathcal{J}}$.

Note that, due to the syntactic restrictions imposed on the uses of PFD constructors, a negation of an PFD can be enforced only in the counterexample of the description E. Spurious occurrences of negated PFDs in the interpretation $\mathcal{I}$ are therefore ignored as the interpretation itself satisfies all PFDs in $\mathcal{T}$.

It is easy to verify that $\delta o \in (\mathsf{Nd}^0 \sqcap \mathrm{D}^0 \sqcap \mathrm{Not(E)})^{\mathcal{J}}$ for $o \in (\mathrm{D} \sqcap \neg \mathrm{E})^{\mathcal{I}}$. By inspecting all inclusion dependencies in $\mathcal{T}$ we have $\mathcal{J} \models \mathcal{T}^i$ as $\mathcal{I} \models \mathcal{T}$. Furthermore, the construction of $\mathcal{J}$ enforces $\mathcal{J} \models \mathcal{E}(l) \cup \mathcal{N}(l)$.

On the other hand, given an interpretation $\mathcal{J}$ of $(\mathsf{Nd}^0 \sqcap \mathrm{D}^0 \sqcap \mathrm{Not(E)})$ that satisfies all assertions in

$$(\bigcup_{0 \le i \le l} \mathcal{T}^i) \cup (\bigcup_{0 \le i < j \le l} \mathcal{T}^{i,j}) \cup \mathcal{E}(l) \cup \mathcal{N}(l),$$

we construct an interpretation $\mathcal{I}$ of $\mathcal{T}$ that falsifies $\mathrm{D} \sqsubseteq \mathrm{E}$ as follows:

- $\Delta^{\mathcal{I}} = \{(o, i) : o \in (\mathsf{Nd}^i)^{\mathcal{J}}, 0 \le i \le l$ and $o \notin (\mathsf{Eq}^{j,i})^{\mathcal{J}}$ for any $0 \le j < i\}$,

- $(f)^{\mathcal{I}}((o,i)) = (o',j)$ whenever $(f)^{\mathcal{J}}(o) = o'$ where j is the smallest integer such that $o \in (\mathsf{Eq}^{j,i})^{\mathcal{J}}$ if such value exists and i otherwise; and
- $(o,i) \in (\mathrm{D})^{\mathcal{I}}$ whenever $(o,i) \in \Delta^{\mathcal{J}}$ and $o \in (\mathrm{D}^i)^{\mathcal{J}}$.

It is easy to verify that $(o,0)$ falsifies $\mathrm{D} \sqsubseteq \mathrm{E}$ whenever o belongs to $(\mathsf{Nd}^0 \sqcap \mathrm{D}^0 \sqcap \mathrm{Not(E)})$, and such an object must exist by our assumptions. Also, $\mathcal{I} \models \mathcal{T}$, as otherwise by cases analysis we get a contradiction with $\mathcal{J} \models (\bigcup_{0 \leq i \leq l} \mathcal{T}^i) \cup (\bigcup_{0 \leq i < j \leq l} \mathcal{T}^{i,j}) \cup \mathcal{E}(l) \cup \mathcal{N}(l)$. □

Corollary 11 *The implication problem for limited $\mathcal{DLFD}$ is decidable and EXPTIME-complete.*

Proof: Follows immediately from Theorems 7 and 10 above. □

5 Conclusions

We have shown that allowing PFDs to occur in the scope of negation or on the left hand sides of inclusion dependencies in the DL $\mathcal{DLFD}$ leads to undecidability of its logical implication problem, and therefore that a full integration of keys and functional dependencies in expressive DLs is not in general possible. Conversely, by virtue of reductions to simpler dialects, we have shown that the complexity of this problem remains unchanged for limited $\mathcal{DLFD}$ in which PFDs are restricted to occur within the scope of monotone concept constructors on right hand sides of inclusion dependencies.

There are several ways that limited $\mathcal{DLFD}$ can be extended without changing the complexity of its logical implication problem. For example, by using reductions introduced in [16], it is straightforward to add roles, quantified number restrictions on roles and even role inversion. (Feature inversion, however, is another matter since its addition to simple $\mathcal{DLFD}$ already leads to undecidability [15, 17].)

There is also a possibility of extending limited $\mathcal{DLFD}$ with *regular path functional dependencies* as defined in [18]. In this case, left and right-hand-sides of PFDs are specified as regular languages that can define infinite sets of path functions. Such constraints have applications in reasoning about equality in semistructured databases [18] and in capturing inductive data types in information integration, thus extending the work in [11].

Another direction of future research includes studying terminologies stratified with respect to the interactions of the PFD constructor and negation in an attempt to extend the applicability of the proposed approach.

References

[1] R. Berger. The undecidability of the dominoe problem. *Mem. Amer. Math. Soc.*, 66:1–72, 1966.

[2] Joachim Biskup and Torsten Polle. Decomposition of Database Classes under Path Functional Dependencies and Onto Constraints. In *Foundations of Information and Knowledge Systems*, pages 31–49, 2000.

[3] Joachim Biskup and Torsten Polle. Adding inclusion dependencies to an object-oriented data model with uniqueness constraints. *Acta Informatica*, 39:391–449, 2003.

[4] Alexander Borgida and Grant E. Weddell. Adding Uniqueness Constraints to Description Logics (Preliminary Report). In *International Conference on Deductive and Object-Oriented Databases*, pages 85–102, 1997.

[5] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 155–160, 2001.

[6] Diego Calvanese, Maurizio Lenzerini, and Giuseppe De Giacomo. Keys for Free in Description Logics. In *Proceeding of the 2000 International Workshop on Description Logics*, pages 79–88, 2000.

[7] David DeHaan, David Toman, and Grant E. Weddell. Rewriting Aggregate Queries using Description Logics. In *Description Logics 2003*, pages 103–112. CEUR-WS vol.81, 2003.

[8] Minoru Ito and Grant E. Weddell. Implication Problems for Functional Constraints on Databases Supporting Complex Objects. *Journal of Computer and System Sciences*, 49(3):726–768, 1994.

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

[10] Vitaliy L. Khizder, David Toman, and Grant E. Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *International Conference on Database Theory ICDT'01*, pages 54–67, 2001.

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

[12] C. Lutz and M. Milicic. Description Logics with Concrete Domains and Functional Dependencies. In *Proc. European Conference on Artificial Intelligence (ECAI)*, pages 378–382, 2004.

[13] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, Nominals, and Concrete Domains. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 349–354, 2003.

[14] Lubomir Stanchev and Grant E. Weddell. Index Selection for Embedded Control Applications using Description Logics. In *Description Logics 2003*, pages 9–18. CEUR-WS vol.81, 2003.

[15] David Toman and Grant Weddell. On the Interaction between Inverse Features and Path-functional Dependencies in Description Logics. In *To appear: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2005.

[16] David Toman and Grant E. Weddell. On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem. In *Description Logics 2001*, pages 76–85. CEUR-WS vol.49, 2001.

[17] David Toman and Grant E. Weddell. Attribute Inversion in Description Logics with Path Functional Dependencies. In *Description Logics 2004*, pages 178–187. CEUR-WS vol.104, 2004.

[18] David Toman and Grant E. Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 2004. doi:10.1016/j.tcs.2004.10.036.

[19] P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture notes in pure and applied mathematics*, pages 331–363. Marcel Dekker Inc., 1997.

[20] 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.