On Keys and Functional Dependencies as First-class Citizens in Description Logics

David Toman and Grant Weddell

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

Abstract. We investigate whether *identification constraints* such as keys and functional dependencies can be granted full status as a concept constructor in a Boolean-complete description logic. In particular, we show that surprisingly simple forms of such constraints lead to undecidability of the associated logical implication problem if they are allowed within the scope of a negation or on the left-hand-side of inclusion dependencies. We then show that allowing a very general form of identification constraints to occur in the scope of *monotone* concept constructors on the right-hand-side of inclusion dependencies still leads to decidable implication problems. Finally, we consider the relationship between certain classes of identification constraints and nominals.

1 Introduction

To date, description logics (DLs) have incorporated keys or functional dependencies in one of two ways. The first adds a separate family of terminological constraints to inclusion dependencies, e.g., in the form of a *key box* [5, 7, 13, 14], while the second avoids this by adding a new concept constructor called a *path-functional dependency* (PFD) [11, 18, 19]. However, the latter approach still falls short of a full integration of keys or functional dependencies since there are syntactic restrictions on occurrences of PFDs and on the syntax of the PFD constructor itself. In particular, all earlier work has required that any occurrence of this constructor appears only at the top level on the right hand side of inclusion dependencies, and that the left hand sides of PFDs themselves are nonempty . Note that an ordinary functional dependency of the form "$\{\} \rightarrow A$" has an empty left hand side and consequently enforces a fixed A value. In this paper, we investigate whether such syntactic restrictions are necessary—unfortunately, it turns out that this is indeed the case—and study the limits of decidability in such a setting. Our main contributions are as follows.

– We show that allowing PFDs on the left hand side of inclusion dependencies, or in the scope of a containing negation on the right hand side, leads to undecidability. Notably, this remains true when PFDs are limited to very simple forms of relational keys or functional dependencies.

– Conversely, we show that allowing PFDs within the scope of *monotone* concept constructors on the right hand side of inclusion dependencies, still leads to decidable implication problems.
– We show that allowing left hand sides of PFDs to be empty also leads to undecidability. This entails first showing that the introduction of an ABox to previously decidable PFD dialects already makes logical implication undecidable. The result follows by showing that such PFDs can simulate nominals.

DLs have become an important part of the semantic web. Indeed, OWL, the current standard for capturing semantic web ontologies, is largely based on a DL dialect. They have also been used as a lingua franca for a large variety of languages for capturing metadata: UML class diagrams, ER models, relational schema, object-oriented schema, DTDs for XML and XML itself are all examples [6, 15].

1.1 Identification is Important

Identification constraints are fundamentally tied to issues of equality, and as web services with query languages such as SWRL that are based on OWL are introduced, important questions such as *how can one reliably identify resources* and *whether there is at most one kind of web service* inevitably surface when finding execution strategies for services and when communicating results of such services. With the addition of the PFD concept constructor along the lines considered in this paper, it becomes possible to express, e.g., that *among all possible clients, social security numbers are a reliable way of identifying those that are registered.* In particular, this can be captured by the following inclusion dependency.

$$\text{Client} \sqsubseteq \neg\text{Registered} \sqcup \text{Client} : \mathit{SIN} \rightarrow \mathit{Id}$$

To paraphrase: *If a client is registered, then no other client will share his or her social insurance number.* Note that social insurance numbers may not be a reliable way of identifying an arbitrary unregistered client in general.

There are a number of additional applications and capabilities that become possible after removing syntactic restrictions on PFDs, beyond the fact that a simple and elegant presentation of the associated DL would ensue. For example, to say that *all information about clients is located at a single site*, one can add the dependency

$$\text{Client} \sqsubseteq \text{Client} :\rightarrow \mathit{LocationOfData}$$

Again to paraphrase: *For any pair of clients, both will agree on the location of available data.*

As we shall see, relaxing existing restrictions to accommodate the first example is possible since disjunction is a monotone concept constructor, but is not possible for the second example without the introduction of alternative restrictions on the use of PFDs or syntax of the PFD constructor itself.

1.2 Background and Related Work

PFDs were introduced and studied in the context of object-oriented data models [9, 23]. An FD concept constructor was proposed and incorporated in Classic [4], an early DL with a PTIME reasoning procedure, without changing the complexity of its implication problem. The generalization of this constructor to PFDs alone leads to EXPTIME completeness of the implication problem [11]; this complexity remains unchanged in the presence of additional concept constructors common in rich DLs [18, 19]. Note that all earlier work has assumed the above syntactic restrictions on occurrences of the PFD concept constructor in inclusion dependencies.

In [5], the authors consider a DL with functional dependencies and a general form of keys added as additional varieties of dependencies, called a *key box*. They show that their dialect is undecidable for DLs 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 [17]. We therefore disallow inverse features in this paper to exclude an already known cause for undecidability.

A form of key dependency with left hand side feature paths has been considered for a DL coupled with various concrete domains [14, 13]. In this case, the authors explore how the complexity of satisfaction is influenced by the selection of a concrete domain together with various syntactic restrictions on the key dependencies themselves.

PFDs have also been used in a number of applications in object-oriented schema diagnosis and synthesis [2, 3], in query optimization [8, 10] and in the selection of indexing for a database [16]. The results reported in this paper are an expansion of earlier preliminary work in [20].

The remainder of the paper is organized as follows. The definition of $\mathcal{DLFD}$, a Boolean complete DL based on attributes or 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 for a variety of simple cases of PFDs. 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 attribute restriction. In Section 5, we consider relaxing the requirement that PFDs have non-empty left hand sides, showing that both this and a (weaker) alternative of admitting an ABox leads to undecidability. Our summary comments follow in Section 6.

2 Definitions

Our investigations are based on the following dialect of description logic called $\mathcal{DLFD}$. To simplify the presentation, the dialect is based on *attributes* or *features* instead of the more common case of roles. Note that $\mathcal{ALCN}$ with a suitable PFD construct can simulate our dialect. Conversely, $\mathcal{DLFD}$ can simulate $\mathcal{ALCQI}$ [21].

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

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

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 fourth production of this grammar is called an* attribute value restriction*. A concept description obtained by using the final production is called a* path functional dependency *(PFD). Note that we assume for this production that* $k > 0$*, that the left hand side of a PFD is non-empty. In addition, a PFD is called: (1)* unary *when* $k = 1$*, (2)* key *when the right hand side is* Id*, and (3)* simple *when there is no path expression with more than a single attribute name.*

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 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

It turns out that allowing arbitrary use of very simple varieties of the PFD concept constructor lead to undecidable implication problems. This is true for three *boundary* cases in particular:

1. when all PFDs are simple and key,
2. when all PFDs are simple and unary, and
3. when all PFDs are simple and non-key.

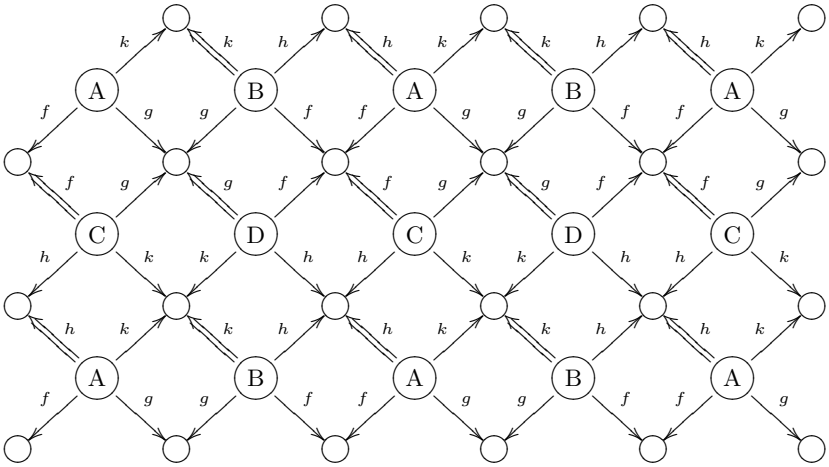

Fig. 2. Defining a Grid.

In the first case, a PFD resembles $C : f_1, \ldots, f_k \rightarrow Id$, which captures the standard notion of *relational keys*, while in the second, a PFD has either the form $C : f \rightarrow g$ or the form $C : f \rightarrow id$. The standard notion of a (relational) *functional dependency* (FD) is captured by the final third case in which a PFD resembles $C : f_1, \ldots, f_k \rightarrow f$.

Observe that the three cases are exhaustive in the sense that the only possibility not covered happens when all PFDs have the form $C : f \rightarrow Id$ (i.e., are simple, unary, and key). However, it is a straightforward exercise in this case to map logical implication problems to alternative formulations in decidable DL dialects with inverses and number restrictions. In the rest of this section, we elaborate on the first two of these cases. Notably, our reductions make no use of attribute value restrictions; they rely solely on PFDs and the standard Boolean constructors. The reduction for the last case is similar and will be given in an extended version of the paper.

Our undecidability results are all based on a reduction of the unrestricted tiling problem to the $\mathcal{DLFD}$ implication problem. An instance U of this 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, 22].

3.1 PFDs that are Simple and Key (relational keys)

The reduction constructs a terminology for a given tiling problem $U = (T, H, V)$, denoted $\mathcal{T}_U^1$, by first establishing an *integer grid* in three steps.

1. Begin by introducing primitive concepts A, B, C and D to serve as possible grid points.

$$\mathrm{A} \sqcap \mathrm{B} \sqsubseteq \bot \quad \mathrm{A} \sqcap \mathrm{C} \sqsubseteq \bot \quad \mathrm{A} \sqcap \mathrm{D} \sqsubseteq \bot \quad \mathrm{B} \sqcap \mathrm{C} \sqsubseteq \bot \quad \mathrm{B} \sqcap \mathrm{D} \sqsubseteq \bot \quad \mathrm{C} \sqcap \mathrm{D} \sqsubseteq \bot$$

2. Then create an "infinitely branching" tree of squares. Such a tree can be rooted at a hypothetical top-left with, e.g., an A object.

$$\begin{array}{l} \mathrm{A} \sqsubseteq \neg(B : g, k \rightarrow Id) \sqcap \neg(C : f, g \rightarrow Id) \\ \mathrm{B} \sqsubseteq \neg(A : f, h \rightarrow Id) \sqcap \neg(D : f, g \rightarrow Id) \\ \mathrm{C} \sqsubseteq \neg(A : h, k \rightarrow Id) \sqcap \neg(D : g, k \rightarrow Id) \\ \mathrm{D} \sqsubseteq \neg(B : h, k \rightarrow Id) \sqcap \neg(C : f, h \rightarrow Id) \end{array}$$

3. And finally, flatten and align the tree into an integer grid using keys.

$$\mathrm{A} \sqsubseteq A : h \rightarrow Id \qquad \mathrm{B} \sqsubseteq B : k \rightarrow Id \qquad \mathrm{C} \sqsubseteq C : f \rightarrow Id \qquad \mathrm{D} \sqsubseteq D : g \rightarrow Id$$

The accumulated effect of these inclusion dependencies on an interpretation is illustrated in Figure 2. Note that the *thick edges* indicate places where flattening of the infinitely branching tree of squares happens.

We model the tiling problem U using primitive concepts T_i for each tile type $t_i \in T$, asserting that $\mathrm{T}_i \sqcap \mathrm{T}_j \sqsubseteq \bot$ for all $i < j$. The tiles are placed on the grid points using the assertion $(\mathrm{A} \sqcup \mathrm{B} \sqcup \mathrm{C} \sqcup \mathrm{D}) \sqsubseteq \bigsqcup_{t_i \in T} \mathrm{T_i}$. The adjacency rules for the instance U of the tiling problem can now be captured as follows:

– for $(t_i, t_j) \notin H$:

$$\begin{array}{ll} \mathrm{A} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{B} \sqcap \mathrm{T}_j) : g \rightarrow Id & \mathrm{B} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{A} \sqcap \mathrm{T}_j) : f \rightarrow Id \\ \mathrm{C} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{D} \sqcap \mathrm{T}_j) : k \rightarrow Id & \mathrm{D} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{C} \sqcap \mathrm{T}_j) : h \rightarrow Id \end{array}$$

– for $(t_i, t_j) \notin V$:

$$\begin{array}{ll} \mathrm{A} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{C} \sqcap \mathrm{T}_j) : f \rightarrow Id & \mathrm{C} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{A} \sqcap \mathrm{T}_j) : h \rightarrow Id \\ \mathrm{B} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{D} \sqcap \mathrm{T}_j) : g \rightarrow Id & \mathrm{D} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{B} \sqcap \mathrm{T}_j) : k \rightarrow Id \end{array}$$

where T_i corresponds to tile type $t_i \in T$. The above constraints form a terminology $\mathcal{T}_U^1$ associated with an unrestricted tiling problem U that immediately yields the following result:

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

$$\mathcal{T}_U^1 \not\models \mathrm{A} \sqsubseteq \bot.$$

Corollary 3 *The logical implication problem for $\mathcal{DLFD}$ with PFDs that are simple and key is undecidable. This remains true in the absence of attribute values restrictions.*

3.2 PFDs that are Simple and Unary

Again, the reduction constructs a terminology for a given tiling problem $U = (T, H, V)$, denoted this time as $\mathcal{T}_U^2$, by first establishing an integer grid. For this case, an extra step is needed.

1. As before, introduce primitive concepts A, B, C and D to serve as possible grid points.

$$\mathrm{A} \sqcap \mathrm{B} \sqsubseteq \bot \quad \mathrm{A} \sqcap \mathrm{C} \sqsubseteq \bot \quad \mathrm{A} \sqcap \mathrm{D} \sqsubseteq \bot \quad \mathrm{B} \sqcap \mathrm{C} \sqsubseteq \bot \quad \mathrm{B} \sqcap \mathrm{D} \sqsubseteq \bot \quad \mathrm{C} \sqcap \mathrm{D} \sqsubseteq \bot$$

2. To create an analogous infinitely branching tree of squares, first create "raw material" consisting of the necessary chains.

$$\mathrm{A} \sqsubseteq \neg(B : k \to Id) \sqcap \neg(C : f \to Id) \quad \mathrm{B} \sqsubseteq \neg(A : h \to Id) \sqcap \neg(D : g \to Id)$$
$$\mathrm{C} \sqsubseteq \neg(A : h \to Id) \sqcap \neg(D : g \to Id) \quad \mathrm{D} \sqsubseteq \neg(B : k \to Id) \sqcap \neg(C : f \to Id)$$

3. The chains are then shaped into squares with functional dependencies.

$$\mathrm{A} \sqsubseteq (B : k \to g) \sqcap (C : f \to g) \quad \mathrm{B} \sqsubseteq (A : h \to f) \sqcap (D : g \to f)$$
$$\mathrm{C} \sqsubseteq (A : h \to k) \sqcap (D : g \to k) \quad \mathrm{D} \sqsubseteq (B : k \to h) \sqcap (C : f \to h)$$

4. And then as before, an integer grid is obtained by flattening and aligning using keys.

$$\mathrm{A} \sqsubseteq A : h \to Id \quad \mathrm{B} \sqsubseteq B : k \to Id \quad \mathrm{C} \sqsubseteq C : f \to Id \quad \mathrm{D} \sqsubseteq D : g \to Id$$

The final accumulated effect of these inclusion dependencies on an interpretation is the same as in Figure 2, and, since a tiling problem can then be overlaid on this grid in the same manner as in the above case for PFDs that are simple and key, we have the following result:

Theorem 4 *An instance $U = (T, H, V)$ of the infinite tiling problem admits a solution if and only if*

$$\mathcal{T}_U^2 \not\models \mathrm{A} \sqsubseteq \bot.$$

Corollary 5 *The logical implication problem for $\mathcal{DLFD}$ with PFDs that are simple and unary is undecidable. This remains true in the absence of attribute value restrictions.*

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.

$$D ::= C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \mid \neg D$$
$$E ::= D \mid E_1 \sqcap E_2 \mid E_1 \sqcup E_2 \mid \forall f.E \mid D : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \to \mathsf{Pf}$$

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 6 $\mathrm{D}_1 \sqsubseteq \neg(\mathrm{D}_2 : f \to g)$ is equivalent to $\mathrm{D}_1 \sqcap (\mathrm{D}_2 : f \to g) \sqsubseteq \bot$.

In the following, we reduce logical implication problems in limited $\mathcal{DLFD}$ to simpler formulations for which existing decisions procedures can be applied [18].

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 7 (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.*

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

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:

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

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

Lemma 8 *1. Let* $\mathcal{I} \models \mathcal{T}^{\mathrm{simp}}$. *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* $\mathcal{I}' \models \mathcal{T}^{\mathrm{simp}}$.

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

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 9 *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 8 and of reductions presented in [18]. □

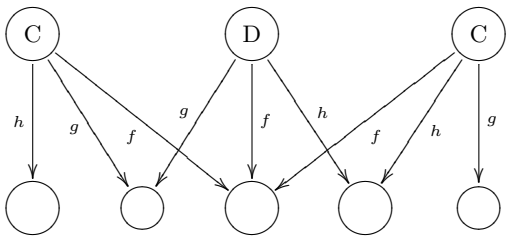

Fig. 3. Counterexample for Example 10.

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 as the following two examples illustrate.

Example 10 A counterexample to $\mathrm{D} \sqsubseteq (\mathrm{C} : f, g \rightarrow h) \sqcup (\mathrm{C} : f, h \rightarrow g)$ is depicted in Figure 3. Note that any such counterexample must also falsify $\mathrm{C} \sqsubseteq \mathrm{C} : f \rightarrow \mathit{Id}$ since distinct C objects that agree on f will be required. Thus:

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

Example 11 A counterexample to $\mathrm{D} \sqsubseteq (\mathrm{C} : f \rightarrow g) \sqcup \forall f.(\mathrm{C} : f \rightarrow g)$ is shown in Figure 4. Observe with this case that distinct C objects must occur at *different* levels when compared to a D-rooted tree.

The examples suggest 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

$$\mathcal{T}^i = \{\mathsf{Nd}^i \sqcap \mathrm{D}^i \sqsubseteq \mathrm{E}^i \mid \mathrm{D} \sqsubseteq \mathrm{E} \in \mathcal{T}, E \text{ a non PFD}\}, \text{ and}$$

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

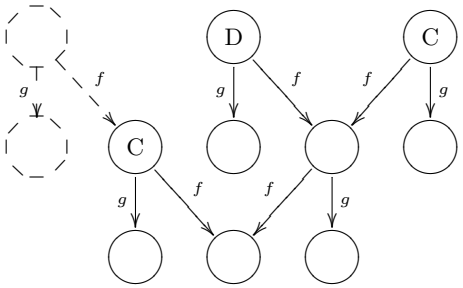

Fig. 4. Counterexample for Example 11.

For a concept description E we define

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

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 $\mathsf{Eq}^{i,j}, 0 \le i \ne j \le \ell$, to express that the ith and jth object copies coincide, and $\mathsf{Nd}^i, 0 \le i \le \ell$, 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.

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

Theorem 12 *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 those 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 $0 \leq i \neq j \leq l$ 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
- $\delta o \in (\mathsf{Nd}^i)^{\mathcal{J}} \wedge o \in (\neg \mathrm{D} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf})^{\mathcal{I}}$ where $\mathrm{D} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$ 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 $\delta o' \in (\mathsf{Nd}^i \sqcap \mathrm{D}^i \sqcap (\sqcap_{1 \leq i \leq k} \forall\, \mathsf{Pf}_i \,.\mathsf{Eq}^{0,i}) \sqcap \forall\, \mathsf{Pf} \,.\neg \mathsf{Eq}^{0,i})^{\mathcal{J}}$.

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

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

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

- $\Delta^{\mathcal{I}} = \{(o, i) : o \in (\mathsf{Nd}^i)^{\mathcal{J}}, 0 \leq i \leq l \text{ and } o \notin (\mathsf{Eq}^{j,i})^{\mathcal{J}} \text{ for any } 0 \leq j < i\}$,
- $(f)^{\mathcal{I}}((o, i)) = (o', j)$ 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 13 *The implication problem for limited $\mathcal{DLFD}$ is decidable and EXPTIME-complete.*

Proof: Follows immediately from Theorems 9 and 12 above. □

5 On PFDs, Nominals and ABoxes

In this section, we explore the possibility of relaxing the non-emptiness condition for left hand sides of PFDs in limited $\mathcal{DLFD}$. Doing so is highly desirable, since, as we have hinted in our introductory comments, this would effectively endow limited $\mathcal{DLFD}$ with a capability for *nominals*. To see this, consider that our introductory *single site for client information* example can be elaborated as follows.

$$\text{Site3} \sqsubseteq \text{Site3} :\rightarrow \mathit{Id} \quad \top \sqsubseteq \forall \mathit{Site3Ref}.\text{Site3} \quad \text{Client} \sqsubseteq \forall \mathit{LocationOfData}.\text{Site3}$$

The first pair of inclusion dependencies define an individual called Site3 in two steps: (1) establish that at most one such individual exists, and (2) that at least one exists if anything exists. The final inclusion dependency then asserts that the location of data for any given client is this individual, thus accomplishing the objectives.

Although desirable, we show in the remainder of this section that allowing PFDs with empty left hand sides leads to undecidability of the logical implication problem for limited $\mathcal{DLFD}$. To do so, we digress to consider a weaker alternative in which the problem is considered in the context of an ABox, showing for this case to begin with that the problem already becomes undecidable by presenting a reduction of the unrestricted tiling problem. However, it is possible with enough restrictions on the use of other concept constructors to re-obtain decidability [9].

An ABox consists of a finite collection of assertions $\mathcal{A}$ about individuals Ind_1, Ind_2, etc., that denote elements of the domain. An assertion establishes concept membership for individuals with the form "$\text{D}(\text{Ind}_i)$", or attribute values for individuals with the form "$f(\text{Ind}_i, \text{Ind}_j)$". The reduction of a given tiling problem $U = (T, H, V)$ to a logical implication problem for limited $\mathcal{DLFD}$ in the context of an ABox constructs a terminology and ABox pair, denoted $\langle \mathcal{T}_U, \mathcal{A}_U \rangle$, by first establishing an integer grid in steps.

1. Begin by defining a triangle *seed* pattern by including in $\mathcal{A}_U$ the following assertions.

$$\text{X}(\text{Ind}_1) \quad \text{Y}(\text{Ind}_2) \quad \text{Z}(\text{Ind}_3) \quad f(\text{Ind}_1, \text{Ind}_2) \quad g(\text{Ind}_1, \text{Ind}_3) \quad g(\text{Ind}_2, \text{Ind}_3)$$

2. Now use the triangle seed pattern to create a possibly infinite *horizontal sequence* of objects that are instances of alternating concepts A_0 and B_0. The results of this step are illustrated along the top of Figure 5.

$$\begin{array}{rl} & \text{X} \sqsubseteq \text{A}_0 \qquad \text{Y} \sqsubseteq \text{B}_0 \\ \text{A}_0 \sqsubseteq \text{A} \sqcap (\forall f.\text{B}_0) \sqcap (\text{B}_0 : g \rightarrow fh) & \qquad \text{B}_0 \sqsubseteq \text{B} \sqcap (\forall f.\text{A}_0) \sqcap (\text{A}_0 : h \rightarrow fg) \end{array}$$

3. And finally, extend this sequence in the vertical direction to form the integer grid. The results of this step are also illustrated in Figure 5.

$$\text{A} \sqsubseteq (\forall k.\text{A}) \sqcap (\text{B} : g \rightarrow kg) \qquad \text{B} \sqsubseteq (\forall k.\text{B}) \sqcap (\text{A} : h \rightarrow kh)$$

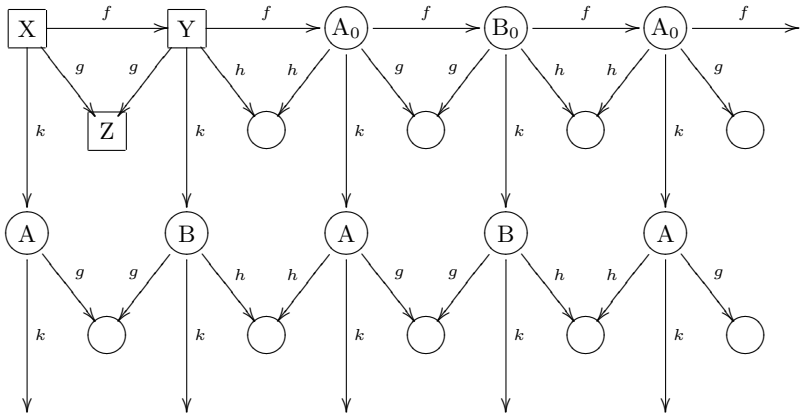

Fig. 5. Defining a Grid using an ABox.

As before, the tiling problem U is modeled using primitive concepts T_i for each tile $t_i \in T$, asserting that $\mathrm{T}_i \sqcap \mathrm{T}_j \sqsubseteq \bot$ for all $i < j$. We place the tiles on the grid points using the assertion $(\mathrm{A} \sqcup \mathrm{B}) \sqsubseteq \bigsqcup_{t_i \in T} \mathrm{T}_i$. The adjacency rules for the instance U of the tiling problem are then captured as follows:

– for $(t_i, t_j) \notin H$:

$$\mathrm{A} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{B} \sqcap \mathrm{T}_j) : g \rightarrow Id \qquad \mathrm{B} \sqcap \mathrm{T}_i \sqsubseteq (\mathrm{A} \sqcap \mathrm{T}_j) : h \rightarrow Id$$

– for $(t_i, t_j) \notin V$:

$$(\mathrm{A} \sqcap \mathrm{T}_i) \sqcap \forall k.(\mathrm{A} \sqcap \mathrm{T}_j) \sqsubseteq \bot \qquad (\mathrm{B} \sqcap \mathrm{T}_i) \sqcap \forall k.(\mathrm{B} \sqcap \mathrm{T}_j) \sqsubseteq \bot$$

where T_i corresponds to a tile type $t_i \in T$. The above constraints form a terminology $\mathcal{T}_U$ and ABox $\mathcal{A}_U$ associated with an unrestricted tiling problem U that immediately yields the following result:

Theorem 14 *An instance $U = (T, H, V)$ of the infinite tiling problem admits a solution if and only if*

$$\langle \mathcal{T}_U, \mathcal{A}_U \rangle \not\models \mathrm{X} \sqsubseteq \bot.$$

Corollary 15 *The logical implication problem for limited $\mathcal{DLFD}$ in the context of an ABox is undecidable.*

Our main result in this section now follows since PFDs with empty left hand sides can simulate the above triangle seed by instead adding the following to $\mathcal{T}_U$:

$$\mathrm{X} \sqcap \mathrm{Y} \sqsubseteq \bot \quad \mathrm{X} \sqsubseteq (\forall f.\mathrm{Y}) \sqcap (\forall g.\mathrm{Z}) \quad \mathrm{Y} \sqsubseteq \forall g.\mathrm{Z} \quad \mathrm{Z} \sqsubseteq \mathrm{Z} :\rightarrow Id$$

Corollary 16 *The logical implication problem for limited $\mathcal{DLFD}$ in which PFDs are permitted empty left hand sides is undecidable.*

6 Conclusions

We have shown that allowing PFDs to occur in the scope of negation or on the left hand sides of inclusion dependencies in $\mathcal{DLFD}$ 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 [18], 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 [17, 19].)

There is also a possibility of extending limited $\mathcal{DLFD}$ with *regular path functional dependencies* as defined in [21]. 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 [21] and in capturing inductive data types in information integration, thus extending the work in [12].

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, Giuseppe De Giacomo, and Maurizio Lenzerini. Representing and reasoning on xml documents: A description logic approach. *J. Log. Comput.*, 9(3):295–318, 1999.
7. 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.

8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. U. Sattler, D. Calvanese, and R. Molitor. Relationships with other formalisms. In *The Description Logic Handbook: Theory, Implementation, and Applications*, chapter 4, pages 137–177. Cambridge University Press, 2003.
16. 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.
17. David Toman and Grant Weddell. On the Interaction between Inverse Features and Path-functional Dependencies in Description Logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 603–608, 2005.
18. 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.
19. 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.
20. David Toman and Grant E. Weddell. On Path-functional Dependencies as First-class Citizens in Description Logics. In *Description Logics 2005*. CEUR-WS vol.147, 2005.
21. David Toman and Grant E. Weddell. On Reasoning about Structural Equality in XML: A Description Logic Approach. *Theoretical Computer Science*, 336:181–203, 2005. doi:10.1016/j.tcs.2004.10.036.
22. P. van Emde Boas. The convenience of tilings. In *Complexity, Logic, and Recursion Theory*, volume 187 of *Lecture notes in pure and applied mathe-matics*, pages 331–363. Marcel Dekker Inc., 1997.
23. 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.