# FunDL

## A Family of Feature-Based Description Logics, with Applications in Querying Structured Data Sources

Stephanie McIntyre, David Toman(✉), and Grant Weddell

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

**Abstract.** Feature-based description logics replace the notion of *roles*, interpreted as binary relations, with *features*, interpreted as unary functions. Another notable feature of these logics is their use of path functional dependencies that allow for complex identification constraints to be formulated. The use of features and path functional dependencies makes the logics particularly well suited for capturing and integrating data sources conforming to an underlying object-relational schema that include a variety of common integrity constraints. We first survey expressive variants of feature logics, including the boundaries of decidability. We then survey a restricted tractable family of feature logics suited to query answering, and study the limits of tractability of reasoning.

## 1 Introduction

We survey the work we have done on developing FunDL, a family of description logics that can be used to address a number of problems in querying structured data sources, with a particular focus on data sources that have an underlying object-relational schema. All member dialects of this family have two properties in common. First, each is *feature based*: the usual notion of *roles* in description logic that are interpreted as binary relations is replaced with the notion of features that are interpreted as unary functions. We have found features to be a better fit with object-relational schema, e.g., for capturing the ubiquitous notion of *attributes*. And second, each dialect includes a concept constructor for capturing a variety of equality generating dependencies: so-called *path functional dependencies* (PFDs) that generalize the notions of primary keys, uniqueness constraints and functional dependencies that are again ubiquitous in object-relational schema. PFDs also ensure member dialects do not forgo the ability to capture roles or indeed $n$-ary relations in general. This can be accomplished by the simple expedient of reification via features, and then by employing PFDs to ensure a set semantics for reified relations. Indeed, the dialect $\mathcal{DLFD}$, introduced in the first part of our survey, can capture very expressive role-based dialects of description logics, including dialects with so-called qualified number restrictions, inverse roles, role hierarchies, and so on [29].


C. Lutz et al. (Eds.): Baader Festschrift, LNCS 11560, pp. 404–430, 2019.
https://doi.org/10.1007/978-3-030-22102-7_19

Our survey consists of three general parts, with the first two parts focusing on the problem of logical implication for FunDL dialects with EXPTIME and PTIME complexity, respectively, and in which the dialects assume features are interpreted as *total* functions. In the third part of our survey, we begin with a review of more recent work on how such dialects may be adapted to support features that are instead *partial* functions. We then consider how *role hierarchies* can be captured as concept hierarchies in which the concepts are introduced as reifications of roles. Part three concludes with a review of other reasoning problems, in particular, on knowledge base consistency for FunDL dialects, and on query answering for dialects surveyed in part two.

We begin in the next section with a general introduction to FunDL: what features are, what the various concept constructors are, basic notational conventions, *the grammar protocol we follow to define the various dialects*, and so on. Our survey concludes with a brief overview of related work.

## 2 Background and Definitions

Here, we define a nameless all inclusive member dialect of the FunDL family for the purpose of introducing a space of concept constructors that we then use for defining all remaining dialects in our survey. We also say how a theory is defined by a so-called *terminology* (or TBox) consisting of a finite set of sentences expressing *inclusion dependencies*, and introduce the problem of logical implication of an inclusion dependency by a TBox. Indeed, we focus exclusively on the problem of logical implication throughout the first two parts of our survey.

**Definition 1 (Feature-Based DLs).** Let $\mathsf{F}$ and $\mathsf{PC}$ be sets of feature 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 Fig. 1.

An *inclusion dependency* $\mathcal{C}$ is an expression of the form $C_1 \sqsubseteq C_2$. A *terminology* (TBox) $\mathcal{T}$ consists of a finite set of inclusion dependencies. A *posed question* $\mathcal{Q}$ is a single inclusion dependency.

The *semantics* of expressions is defined with respect to a structure $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$, where $\triangle$ is a domain of objects or entities and $(\cdot)^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concept names $A$ to be subsets of $\triangle$ and feature names $f$ to be total functions $(f)^{\mathcal{I}} : \triangle \to \triangle$. 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 $C$ as defined in the centre column of Fig. 1.

An interpretation $\mathcal{I}$ *satisfies an inclusion dependency* $C_1 \sqsubseteq C_2$ if $(C_1)^{\mathcal{I}} \subseteq (C_2)^{\mathcal{I}}$ and is a *model of* $\mathcal{T}$ ($\mathcal{I} \models \mathcal{T}$) if it satisfies all inclusion dependencies in $\mathcal{T}$. The *logical implication problem* asks if $\mathcal{T} \models \mathcal{Q}$ holds, that is, if $\mathcal{Q}$ is satisfied in all models of $\mathcal{T}$. □

We shall see that the logical implication problem for this logic is undecidable for a variety of reasons. For example, the value restriction, top and same-as concept constructors are all that are needed to encode the uniform word problem [24].

| Syntax | Semantics: Defn of $(\cdot)^{\mathcal{I}}$ | |
|---|---|---|
| $C ::= A$ | $(A)^{\mathcal{I}} \subseteq \triangle$ | (primitive concept; $A \in \mathsf{PC}$) |
| $\mid C_1 \sqcap C_2$ | $(C_1)^{\mathcal{I}} \cap (C_2)^{\mathcal{I}}$ | (conjunction) |
| $\mid \forall \mathsf{Pf}.C$ | $\{x \mid (\mathsf{Pf})^{\mathcal{I}}(x) \in (C)^{\mathcal{I}}\}$ | (value restriction) |
| $\mid C_1 \sqcup C_2$ | $(C_1)^{\mathcal{I}} \cup (C_2)^{\mathcal{I}}$ | (disjunction) |
| $\mid \neg C$ | $\triangle \setminus (C)^{\mathcal{I}}$ | (negation) |
| $\mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}$ | $\{x \mid \forall y \in (C)^{\mathcal{I}}. \bigwedge_{i=1}^{k} (\mathsf{Pf}_i)^{\mathcal{I}}(x) = (\mathsf{Pf}_i)^{\mathcal{I}}(y) \rightarrow (\mathsf{Pf})^{\mathcal{I}}(x) = (\mathsf{Pf})^{\mathcal{I}}(y)\}$ | (PFD) |
| $\mid \exists f^{-1}$ | $\{x \mid \exists y \in \triangle : (f)^{\mathcal{I}}(y) = x\}$ | (feature inverse) |
| $\mid \exists f^{-1}.C$ | $\{x \mid \exists y \in (C)^{\mathcal{I}} : (f)^{\mathcal{I}}(y) = x\}$ | (qualified feature inverse) |
| $\mid \top$ | $\triangle$ | (top) |
| $\mid \bot$ | $\emptyset$ | (bottom) |
| $\mid (\mathsf{Pf}_1 = \mathsf{Pf}_2)$ | $\{x \mid (\mathsf{Pf}_1)^{\mathcal{I}}(x) = (\mathsf{Pf}_2)^{\mathcal{I}}(x)\}$ | (same-as) |

**Fig. 1.** Concept constructors in feature-based description logics.

Thus, each dialect of the FunDL family in our survey will correspond to some fragment of this logic. Grammars defining a dialect use the non-terminals $C$ and $D$ to characterize concept constructors permitted on left-hand-sides and right-hand-sides of inclusion dependencies occurring in a TBox, respectively, and the non-terminal $E$ to characterize concept constructors permitted in posed questions. We also assume, when an explicit definition of non-terminal $D$ (resp. $E$) is missing, that $D$ concept descriptions align with $C$ concept descriptions (resp. $E$ concept descriptions align with $D$ concept descriptions).

To see how FunDL dialects are useful in capturing structured data sources, consider a visualization of a hypothetical object-relational university schema in Fig. 2. Here, nodes are classes, labelled directed edges are attributes, thick edges denote inheritance, and underlined attributes denote primary keys. Introducing a primitive concept and a feature for each class and attribute then enables attribute typing, inheritance, primary keys and a variety of other data dependencies to be captured as inclusion dependencies in a university TBox:

1. (*disjoint classes*) PERSON $\sqsubseteq \neg$DEPT,
2. (*attribute typing*) PERSON $\sqsubseteq \forall$*name*.STRING,
3. (*unary primary key*) PERSON $\sqsubseteq$ PERSON : *name* $\rightarrow$ *id*,
4. (*disjoint attribute values*) PERSON $\sqsubseteq$ DEPT : *name* $\rightarrow$ *id*,
5. (*inheritance*) PROF $\sqsubseteq$ PERSON,
6. (*views*) $\forall$*reports*.CHAIR $\sqsubseteq$ PROF,
7. (*mandatory participation*) $\exists head^{-1} \sqsubseteq$ CHAIR,
8. (*binary primary key*) CLASS $\sqsubseteq$ CLASS : *dept*, *num* $\rightarrow$ *id*, and
9. (*cover*) PERSON $\sqsubseteq$ (STUDENT $\sqcup$ PROF).

Allowing path expressions to occur in PFD concepts turns out to be quite useful in capturing additional varieties of equality generating dependencies, as in the

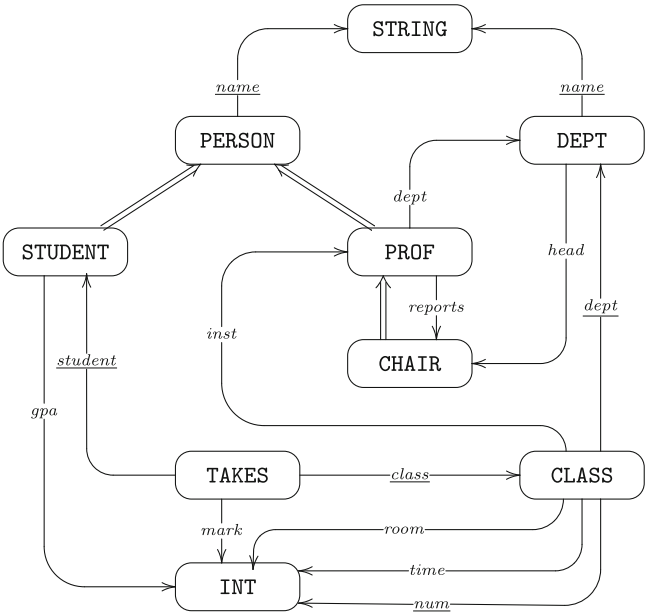


**Fig. 2.** An object-relational schema.

following:

$$\text{TAKES} \sqsubseteq \text{TAKES} : \mathit{student}, \mathit{class.room}, \mathit{class.time} \rightarrow \mathit{class}.$$

This inclusion dependency expresses a constraint induced by the interaction of time and space, that *no student can take two different classes in the same room at the same time* or, to paraphrase, that *no pair of classes with at least one student taking them can be in the same room at the same time*. The second reading illustrates how so-called *identification constraints* in DL-Lite dialects can also be captured [11].

In the third part of our survey, we review work on how features may be interpreted as partial functions. This leads to the addition of the concept constructor $\exists f$ for capturing domain elements for which feature $f$ is defined. Consequently, it becomes possible to say, e.g., that a DEPT does not have a *gpa* by adding the inclusion dependency

$$\text{DEPT} \sqsubseteq \neg\exists \mathit{gpa}$$

to the university TBox.

Note that any logical implication problem for the university TBox defined thus far can be solved by appeal to one of the expressive FunDL dialects, and, notwithstanding *cover* constraints, can be solved by one of the tractable dialects in PTIME. An ability to do this has many applications in information systems technology. For example, early work on FunDL has shown how to reduce the

problem of determining when a SQL query can be reformulated without mentioning the DISTINCT keyword to a logical consequence problem [20]. More recent applications allow one to resolve fundamental issues in reasoning about identity in conceptual modelling and SQL programming [6], and in ontology-based data access [7,26].

### 2.1 Ackerman Decision Problems

Our complexity reductions are tied to the classical *Ackermann case of the decision problem* [1].

**Definition 2 (Monadic Ackerman Formulae).** Let $P_i$ be monadic predicate symbols and $x, y_i, z_i$ variables. A *monadic first-order formula in the Ackermann class* is a formula of the form $\exists z_1 \ldots \exists z_k \forall x \exists y_1 \ldots \exists y_l . \varphi$ where $\varphi$ is a quantifier-free formula over the symbols $P_i$. □

Every formula with the Ackermann prefix can be converted to *Skolem normal form* by replacing variables $z_i$ by Skolem constants and $y_i$ by unary Skolem functions not appearing in the original formula. This, together with standard Boolean equivalences, yields a finite set of universally-quantified clauses containing at most one variable ($x$).

**Proposition 3 ([16]).** The Ackermann decision problem is complete for EXPTIME.

The lower bound holds even for the Horn fragment of the decision problem called Datalog$_{nS}$ [15]. A Datalog$_{nS}$ program is a finite set of *definite Horn* Datalog$_{nS}$ clauses. A *recognition problem* for a Datalog$_{nS}$ program $\Pi$ and a ground atom $Q$ is to determine if $Q$ is true in all models of $\Pi$ (i.e., if $\Pi \cup \{\neg Q\}$ is unsatisfiable).

## 3 Expressive FunDL Dialects

In this first part of our survey, we consider the logical implication problem for an expressive Boolean complete dialect with value restrictions on features. We begin by presenting a lower bound for a fragment of this dialect and then follow with upper bounds. We subsequently consider extensions to the dialect that admit additional concept constructors, namely PFDs and inverse features.

### 3.1 Logical Implication in $\mathcal{DLF}$

The dialect $\mathcal{DLF}_0$ of FunDL is defined by the following grammar (and recall our protocol whereby right-hand-sides of inclusion dependencies and posed questions are also defined by non-terminal $C$):

$$C ::= A \mid C_1 \sqcap C_2 \mid \forall f.C$$

Observe that $\mathcal{DLF}_0$ is a Horn fragment that only allows primitive concepts, conjunctions and value restrictions. We show that every Datalog$_{nS}$ recognition

problem can be simulated by a $\mathcal{DLF}_0$ implication problem [29]. For this reduction, each monadic predicate symbol is assumed to also qualify as a primitive concept name in $\mathcal{DLF}_0$. Given an instance of a Datalog$_{nS}$ recognition problem in the form of a Datalog$_{nS}$ program $\Pi$ and a ground goal atom $G = P(\overline{\mathsf{Pf}}(0))$, we construct an implication problem for $\mathcal{DLF}_0$ as follows: in $\Pi$,

$$\begin{aligned}\mathcal{T}_\Pi \quad &= \{\forall\, \mathsf{Pf}'_1\,.Q'_1 \sqcap \ldots \sqcap \forall\, \mathsf{Pf}'_k\,.Q'_k \sqsubseteq \forall\, \mathsf{Pf}'\,.P' : \\ &\qquad P'(\overline{\mathsf{Pf}}'(x)) \leftarrow Q'_1(\overline{\mathsf{Pf}}'_1(x)), \ldots, Q'_k(\overline{\mathsf{Pf}}'_k(x)) \in \Pi\}, \\ \mathcal{Q}_{\Pi,G} &= \forall\, \mathsf{Pf}_1\,.Q_1 \sqcap \cdots \sqcap \forall\, \mathsf{Pf}_k\,.Q_k \sqsubseteq \forall\, \mathsf{Pf}\,.P,\end{aligned}$$

where the $\overline{\mathsf{Pf}}(x)$ terms in Datalog$_{nS}$ naturally correspond to path functions $\mathsf{Pf}$ in $\mathcal{DLF}_0$, and where the posed question $\mathcal{Q}_{\Pi,G}$ is formed from ground facts $Q_i(\overline{\mathsf{Pf}}_i(0)) \in \Pi$, and the ground goal atom $G = P(\overline{\mathsf{Pf}}(0))$.

**Theorem 4 ([30]).** Let $\Pi$ be a Datalog$_{nS}$ program and $G$ a ground atom. Then

$$\Pi \models G \iff \mathcal{T}_\Pi \models \mathcal{Q}_{\Pi,G}.$$

For the reduction to work, one needs two features. (Unlike the case with $\mathcal{ALC}$ style logics, the problem becomes PSPACE-complete with one feature.) This result was later used to show EXPTIME-hardness for $\mathcal{FL}_0$ [3].

We now show a matching upper bound for the Boolean complete dialect with value restrictions, as defined by the following:

$$C ::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C$$

We first show how the *semantics* of $\mathcal{DLF}$ constructors can be captured by Ackermann formulae: let $C$, $C_1$, and $C_2$ range over concept descriptions and $f$ over attribute names. We introduce a unary predicate subscripted by a description that simulates that description in our reduction:

$$\begin{array}{l}\forall x.(P_C(x) \vee P_{\neg C}(x)), \forall x.\neg(P_C(x) \wedge P_{\neg C}(x)) \\ \forall x.P_{C_1 \sqcap C_2}(x) \leftrightarrow (P_{C_1}(x) \wedge P_{C_2}(x)) \\ \forall x.P_{C_1 \sqcup C_2}(x) \leftrightarrow (P_{C_1}(x) \vee P_{C_2}(x)) \\ \forall x.P_{\forall f.C}(x) \leftrightarrow P_C(f(x))\end{array} \qquad (*)$$

To complete the translation of a $\mathcal{DLF}$ implication problem $\mathcal{T} \models \mathcal{Q}$, for $\mathcal{Q}$ of the form $C \sqsubseteq D$, what remains is the translation of the inclusion dependencies in $\mathcal{T} \cup \{\mathcal{Q}\}$:

- $\Phi_{\mathcal{DLF}} = \bigwedge_{\varphi \in \text{Semantics}(\mathcal{T},\mathcal{Q})} \varphi$,
- $\Phi_{\mathcal{T}} = \bigwedge_{C' \sqsubseteq D' \in \mathcal{T}} \forall x.P_{C'}(x) \rightarrow P_{D'}(x)$, and
- $\Phi_{\mathcal{C}} = P_C(0) \wedge P_{\neg D}(0)$ (a Skolemized negation of the posed question $\mathcal{Q}$),

where Semantics$(\mathcal{T}, \mathcal{Q})$ is the set of all formulae $(*)$ whose subscripts range over concepts and subconcepts that appear in $\mathcal{T} \cup \{\mathcal{Q}\}$.

**Theorem 5 ([30]).** Let $\mathcal{T}$ and $\mathcal{Q} = C \sqsubseteq D$ be a terminology and inclusion dependency in $\mathcal{DLF}$, respectively. Then $\mathcal{T} \models \mathcal{C}$ iff $\Phi_{\mathcal{DLF}} \wedge \Phi_{\mathcal{T}} \wedge \Phi_{\mathcal{Q}}$ is not satisfiable.

Theorems 4 and 5 establish a tight EXPTIME complexity bound for the $\mathcal{DLF}$ logical implication problem.

### 3.2 Adding Path Functional Dependencies to $\mathcal{DLF}$

Allowing unrestricted use of the PFD concept constructor leads to undecidable implication problems, as in the case of a description logic defined by the following grammar:

$$C ::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}$$

This remains true even for very simple varieties of PFD concept constructors.

The undecidability results are based on a reduction of the unrestricted tiling problem [4,5] to the logical implication problem. The crux of the reduction is the use of the PFD constructor under negation or, equivalently, on the left-hand-side of inclusion dependencies. For example, the dependency

$$A \sqsubseteq \neg(B : f, g \rightarrow id)$$

states that, for some $A$ object, there must be a distinct $B$ object that agrees with this $A$ object on features $f$ and $g$, i.e., there must be a square in a model of the above inclusion dependency. Such squares can then be connected into a grid using additional PFDs and the Boolean structure of the logic in a way that enables tiling to be simulated.

This idea can be sharpened to the following three borderline cases, where *simple*, *unary* and *key* refer, respectively, to conditions in which path expressions correspond to individual features or to $id$, in which left-hand-sides of PFDs consist of a single path expression, and in which the right-hand-side is $id$ [35]:

1. PFDs are simple and key, and therefore resemble

$$C : f_1, \ldots, f_k \rightarrow id$$

   (i.e., the standard notion of *relational keys*);
2. PFDs are simple and non-key, and therefore resemble

$$C : f_1, \ldots, f_k \rightarrow f$$

   (i.e., the standard notion of relational *functional dependencies*); and
3. PFDs are simple and unary, and therefore resemble either of the following:

$$C : f \rightarrow g \quad \text{or} \quad C : f \rightarrow id\,.$$

Observe that the three cases are exhaustive: the only possibility not covered happens when all PFDs have the form $C : \mathsf{Pf} \rightarrow id$, i.e., are 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 functional restrictions. Notably, the reductions make no use of attribute value restrictions in the first two of these cases; they rely solely on PFDs and the standard Boolean constructors.

**On Regaining Decidability.** It turns out that undecidability is indeed a consequence of allowing PFDs to occur within the scope of negation (and, as a consequence, all FunDL dialects disallow this possibility). Among the first expressive and decidable dialects is $\mathcal{DLFD}$, the description logic defined by the following grammar rules:

$$\begin{aligned} C &::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C \mid \top \\ D &::= C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf} \end{aligned}$$

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.* (Allowing PFDs on left hand sides is equivalent to allowing PFDs in the scope of negation: $D_1 \sqsubseteq \neg(D_2 : f \rightarrow g)$ is equivalent to $D_1 \sqcap (D_2 : f \rightarrow g) \sqsubseteq \bot$.)

To establish the complexity lower bound, we first study the problem for a subset of $\mathcal{DLFD}$ in which all inclusion dependencies are of the form

$$\top \sqsubseteq \top : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.$$

An implication problem in this subset is called a *PFD membership problem.* It will simplify matters to assume that each monadic predicate symbol $P$ in Datalog$_{nS}$ maps to a distinct feature $p$ in $\mathcal{DLFD}$, and that each such $p$ differs from the attributes corresponding to unary function symbols in Datalog$_{nS}$.

We proceed similarly to the $\mathcal{DLF}$ case: Let $\Pi$ be an arbitrary Datalog$_{nS}$ program and $G = P(\overline{\mathsf{Pf}}(0))$ a ground atom. We construct an implication problem for $\mathcal{DLFD}$ as follows:

$$\begin{aligned} \mathcal{T}_\Pi \quad &= \{\top \sqsubseteq \top : \mathsf{Pf}'_1\,.p'_1, \ldots, \mathsf{Pf}'_k\,.p'_k \rightarrow \mathsf{Pf}'\,.p' : \\ &\qquad P'(\overline{\mathsf{Pf}}'(x)) \leftarrow P'_1(\overline{\mathsf{Pf}}'_1(x)), \ldots, P'_k(\overline{\mathsf{Pf}}'_k(x)) \in \Pi\}, \\ \mathcal{C}_{\Pi,G} &= \top \sqsubseteq \top : \mathsf{Pf}_1\,.p_1, \ldots, \mathsf{Pf}_k\,.p_k \rightarrow \mathsf{Pf}\,.p, \end{aligned}$$

where $P_1(\overline{\mathsf{Pf}}_1(0)), \ldots, P_k(\overline{\mathsf{Pf}}_k(0))$ are the ground facts in $\Pi$.

**Theorem 6 ([30]).** Let $\Pi$ be an arbitrary Datalog$_{nS}$ program and $G = P(\overline{\mathsf{Pf}}(0))$ a ground atom. Then $\Pi \models G \iff \mathcal{T}_\Pi \models \mathcal{C}_{\Pi,G}$.

The reduction establishes another source of EXPTIME-hardness for our $\mathcal{DLFD}$ fragment that originates from the PFDs only.

To establish the upper bound, we reduce logical implication in $\mathcal{DLFD}$ to logical implication in $\mathcal{DLF}$. The reduction is based on the following observations:

1. If the posed question does *not* contain the PFD concept constructor then the implication problem reduces to the implication problem in $\mathcal{DLF}$ since, due to the tree model property of the logic, the PFD inclusion dependencies in the TBox are satisfied vacuously;
2. Otherwise the posed question contains a PFD, e.g., has the form

$$A \sqsubseteq B : \mathsf{Pf}_!, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.$$

To falsify the posed question in this case, we need to construct a model consisting of two trees respectively rooted by $A$ and $B$ that obey the TBox inclusion dependencies, that agree on paths $\mathsf{Pf}_1, \ldots, \mathsf{Pf}_k$ originating from the respective roots, and that disagree on $\mathsf{Pf}$. Since the two trees are identical up to node labels and the agreements always equate corresponding nodes in the two trees, the model can be simulated in $\mathcal{DLF}$ by *doubling* the primitive concepts (one for simulating concept membership in each of the two trees) and by introducing an auxiliary primitive concept to simulate path agreements. This *two trees* idea can then be generalized to account for posed questions having (possibly multiple) PFDs nested in other monotone concept constructors.

The above assumes that PFDs are not nested in other constructors in a TBox; this can be achieved by a simple conservative extension of the given TBox and appropriate reformulation of the posed question [35].

**Theorem 7** ([30])**.** The implication problem for $\mathcal{DLFD}$ can be reduced to an implication problem for $\mathcal{DLF}$ with only a linear increase in size.

Theorems 6 and 7 establish a tight EXPTIME complexity bound for the $\mathcal{DLFD}$ implication problem.

### 3.3 Adding Inverse Features

Allowing right-hand-sides of inclusion dependencies to now employ inverse features together with PFDs, as in $\mathcal{DLFDI}$, a FunDL dialect defined by the following grammar:

$$C ::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C \mid \top$$
$$D ::= C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \mid \exists f^{-1}.C \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}$$

leads immediately to undecidability, similarly to [14]. Again, the reduction is from the unrestricted tiling problem in which an *initial square* is generated by the constraints

$$A \sqsubseteq \exists f^{-1}.B \sqcap \exists f^{-1}.C, \quad B \sqcap C \sqsubseteq \bot, \text{ and } B \sqsubseteq C : f \rightarrow g,$$

and further inclusion dependencies then extend it to a properly tiled grid.

**Theorem 8** ([31]) Logical implication for $\mathcal{DLFDI}$ is undecidable.

**On Regaining Decidability with Inverses.** We review two approaches to restricting either the PFD constructor or the way inverses are allowed to be qualified to regain decidability of the logical implication problem.

*Prefix-restricted PFDs.* The first approach syntactically restricts the PFD constructor as follows:

**Definition 9 [Prefix Restricted Terminologies].** Let $D : \mathsf{Pf}.\mathsf{Pf}_1, \ldots, \mathsf{Pf}.\mathsf{Pf}_k \rightarrow \mathsf{Pf}'$ be an arbitrary PFD where $\mathsf{Pf}$ is the maximal common prefix of the path expressions $\{\mathsf{Pf}.\mathsf{Pf}_1, \ldots, \mathsf{Pf}.\mathsf{Pf}_k\}$. The PFD is *prefix-restricted* if either $\mathsf{Pf}'$ is a prefix of $\mathsf{Pf}$ or $\mathsf{Pf}$ is a prefix of $\mathsf{Pf}'$. □

This condition applies to the argument PFDs occurring in a terminology and strengthens the results in [14]. Note that, because of *accidental common prefixes*, it is not sufficient to simply require that unary PFDs resemble keys since, for example, a $k$-ary PFD $A_1 \sqsubseteq A_2 : f.a_1, \ldots, f.a_k \rightarrow h$ has a logical consequence $A_1 \sqsubseteq A_2 : f \rightarrow h$, thus yielding the ability to construct tiling similar to the one outlined above.

**Theorem 10 ([31]).** Let $\mathcal{T}$ be $\mathcal{DLFDI}$ terminology with prefix-restricted PFDs. Then the implication problem $\mathcal{T} \models \mathcal{Q}$ is decidable and EXPTIME-complete.

*Coherent Terminologies.* The second of our conditions for recovering decidability is to impose a coherency condition on terminologies themselves. The main advantage of this approach is that we thereby regain the ability for unrestricted use of PFDs in terminologies. The disadvantage is roughly that there is a *single use* restriction on using feature inversions in terminologies.

**Definition 11 (Coherent Terminologies).** A terminology $\mathcal{T}$ is *coherent* if

$$\mathcal{T} \models (\exists f^{-1}.D) \sqcap (\exists f^{-1}.E) \sqsubseteq \exists f^{-1}(D \sqcap E)$$

for all descriptions $D, E$ that appear as subconcepts of concepts that appear in $\mathcal{T}$, or their negations. □

Note that we can *syntactically guarantee* that $\mathcal{T}$ is coherent by adding inclusion dependencies of the form $(\exists f^{-1}D) \sqcap (\exists f^{-1}E) \sqsubseteq \exists f^{-1}(D \sqcap E)$ to $\mathcal{T}$ for all concept descriptions $D, E$ appearing in $\mathcal{T}$. This restriction allows us to construct interpretations of non-PFD descriptions in which objects do not have more than one $f$ predecessor (for all $f \in \mathsf{F}$) and thus satisfy all PFDs vacuously.

By restricting logical implication problems for $\mathcal{DLFDI}$ to cases in which terminologies are coherent, it becomes possible to apply reductions to satisfiability problems for Ackerman formulae.

**Theorem 12 ([31]).** Let $\mathcal{T}$ be a coherent $\mathcal{DLFDI}$ terminology. Then the implication problem $\mathcal{T} \models \mathcal{C}$ is decidable and EXPTIME-complete.

Note that *unqualified inverse features of the form* $\exists f^{-1}$ immediately imply coherency. Moreover, one can qualify an $f$ predecessor by concept $C$ by asserting

$$A \sqsubseteq \exists f^{-1}, \quad \forall f.A \sqsubseteq C.$$

Thus, the restriction to *unqualified* inverses does not rule out cases in which qualified inverses might be useful, and avoids the problem of allowing multiple

$f$ predecessors (that could then interact with the PFD constructs). Hence, for the remainder of the survey, we assume unqualified inverse features in FunDL dialects.

### 3.4 Equational Constraints

As pointed out in our introductory comments, allowing equational (same-as) concepts in TBoxes leads immediately to undecidability via a reduction from the uniform word problem [24]. Conversely, allowing equational concepts in *posed questions* extends the capabilities of the logics, in particular allowing for capturing factual assertions (called an ABox, see Sect. 5.3). To this end we introduce the FunDL dialect $\mathcal{DLFDE}$ defined as follows:

$$\begin{aligned} C &::= A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C \mid \top \\ D &::= C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf} \\ E &::= C \mid E_1 \sqcap E_2 \mid \bot \mid \neg E \mid \forall f.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2) \end{aligned}$$

**Undecidability.** It is easy to see that the following two restricted cases have decidable decision problems:

- allowing arbitrary PFDs in terminologies, and
- allowing equational concepts in the posed question.

Unfortunately, the combination of the two cases leads again to undecidability. One can use the equational concept to create a seed square for a tiling problem (although a triangle is actually sufficient in this case, as in $A \sqsubseteq (f.g = g) \sqcap \forall f.B$ [35]) that can then be extended into an infinite grid using PFDs in a TBox (e.g., $A \sqsubseteq (B : g \rightarrow f.h) \sqcap (B : g \rightarrow k.g)$ for the triable seed case), and ultimately to an instance of a tiling problem. Hence:

**Theorem 13 ([35]).** Let $\mathcal{T}$ be a $\mathcal{DLFD}$ terminology and $E$ an equational concept. Then the problem $\mathcal{T} \models E \sqsubseteq \bot$ is undecidable.

**Decidability and a Boundary Condition.** To regain decidability, we restrict the PFD constructor to adhere to a *boundary* condition, in particular, to have either of the following two forms:

- $C : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$; and
- $C : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.f$, for some primitive feature $f$.

We call the resulting fragment $\mathcal{DLFDE}^{-}$. The condition distinguishes, e.g., the PFDs $f \rightarrow id$ and $f \rightarrow g$ from the PFD $f \rightarrow g.f$. Intuitively, a simple saturation procedure that *fires* PFDs on a hypothetical database is now guaranteed to terminate as a consequence.

Notice that the boundary condition still admits PFDs that express arbitrary keys or functional dependencies in the sense of the relational model, including those occurring in all our examples. Thus, restricting PFDs in this manner does not sacrifice any ability to capture database schema for legacy data sources.

**Theorem 14** ([21])**.** Let $\mathcal{T}$ and $\mathcal{T}'$ be respective $\mathcal{DLF}$ and $\mathcal{DLFD}$ terminologies in which the latter contains only PFD inclusion dependencies, and let $E$ be an equational concept. Then there is a concept $E'$ such that

$$\mathcal{T} \cup \mathcal{T}' \models E \sqsubseteq \bot \text{ iff } \mathcal{T} \models (E \sqcap E') \sqsubseteq \bot.$$

Moreover, $E'$ can be constructed from $\mathcal{T}'$ and $E$ effectively and in time polynomial in $|\mathcal{T}'|$.

The *boundary* condition on PFDs is essential for the above theorem to hold. If unrestricted PFDs are combined with either equations or an ABox, there is no limit on the *length* of paths participating in path agreements when measured from an initial object $o \in E \sqcap E'$ in the associated satisfiability problem. Moreover, any minimal relaxation of this condition, i.e., allowing only non-key PFDs of the form $C : f \rightarrow g.h$, already leads to undecidability [32,35]:

**Theorem 15** ([21])**.** $\mathcal{DLFDE}^-$ logical implication and the problem of ABox consistency defined in Sect. 5.3 are decidable and complete for EXPTIME.

The construction essentially generates a pattern (part of a model) that satisfies $E$ (which already contains the effects of all PFDs due to the boundary condition) and then tests if this pattern can be extended to a full model using the decision procedure for $\mathcal{DLF}$. Note also that posed questions containing PFDs can be rewritten to equivalent posed questions replacing the PFDs with their semantic definitions via path agreements and disagreements.

**Inverses.** Finally, we conjecture that adding unqualified inverse constructor to $\mathcal{DLFDE}$ under the restrictions outlined in Sect. 4.4 preserves all the results.

## 4 Tractable FunDL Dialects

In this second part of our survey, we consider the logical implication problem for FunDL dialects for which the logical implication problem can be solved in PTIME. We begin by reviewing $\mathcal{CFD}$, chronologically, the first member of the FunDL family and, so far as we are aware, the first DL dialect to introduce a type constructor, PFDs, for capturing equality generating dependencies [8,20].

Ensuring tractability requires that we somehow evade Theorems 4 and 6. This is generally achieved by requiring a TBox to satisfy the following additional conditions:

1. Interaction between value restrictions and conjunctions on the left-hand-sides of inclusion dependencies must somehow be controlled,
2. Inclusion dependencies must be Horn (which effectively disallows the use of disjunction)[1], and

[1] Allowing the use of conjunction at the top level on the right-hand-side is a simple syntactic sugar.

3. PFDs must satisfy an additional syntactic *boundary* condition in addition to being disallowed on the left-hand-side of inclusion dependencies.

We shall see that violating any of these conditions leads to intractability of logical implication.

### 4.1 Horn Inclusion Dependencies

The first way of limiting the interactions between value restrictions and conjunctions on the left-hand-sides of inclusion dependencies is by simply disallowing value restrictions entirely, and by no longer permitting posed questions to mention either negations or disjunctions. This approach underlies the FunDL dialect called $\mathcal{CFD}$ given by the following grammar:

$$\begin{aligned} C &::= A \mid C_1 \sqcap C_2 \\ D &::= C \mid D_1 \sqcap D_2 \mid \forall f.D \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf} \\ E &::= C \mid \bot \mid E_1 \sqcap E_2 \mid \forall f.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2) \end{aligned}$$

The main idea behind decidability and complexity of the logical implication problem is similar to the idea in Theorem 15. However, we no longer need to use the $\mathcal{DLF}$ decision procedure to verify that the partial model can be completed to a full model since, in $\mathcal{CFD}$, one can always employ complete $\mathsf{F}$-trees whose nodes belong to all primitive concepts (without having to check for their existence [20,36]). Hence, the complexity reduces to the construction of the initial part of the model. This, with the help of the restrictions on $E$ concepts, can be done in PTIME.

**Theorem 16 ([36]).** The logical implication problem for $\mathcal{CFD}$ is complete for PTIME.

The hardness follows from the fact that the PFDs alone can simulate HornSAT.

*Extensions Versus Tractability.* Unfortunately, extending this fragment while maintaining tractability is essentially infeasible. The following table summarizes the effects of allowing additional concept constructors in the TBox on the right-hand-side of inclusion dependencies, reading down, and in the posed question, reading across [36]:

| $\mathcal{T}/\mathcal{Q}$ | $\mathcal{CFD}$ or $\mathcal{CFD}_{\neq,(\neg)}$ | $\mathcal{CFD}_{\neq,\sqcup}$ or $\mathcal{CFD}_{\neg}$ |
|---|---|---|
| $\mathcal{CFD}$ | P-c / in P | P-c / coNP-c |
| $\mathcal{CFD}^{\sqcup}$ | coNP-c / coNP-c | coNP-c / coNP-c |
| $\mathcal{CFD}^{\bot}$ | PSPACE-c / in P | PSPACE-c / coNP-c |
| $\mathcal{CFD}^{\sqcup,\bot}$ | EXPTIME-c / coNP-c | EXPTIME-c / coNP-c |

The complexities listed in the table are with respect to the size of the TBox and the size of the posed question. Note in particular that concept *disjointness*,

in which $\bot$ is allowed on right-hand-sides of inclusion dependencies, leads to PSPACE-completeness. This is due to the need for checking whether a partial model can be completed, which in turn requires testing for reachability in an implicit but exponentially-sized graph.

### 4.2 Value Restrictions Instead of Conjunctions

An alternative that allows us to evade the ramifications of Theorem 4 is disallowing *conjunctions* on the left-hand-sides of inclusion dependencies, yielding the dialect $\mathcal{CFD}_{nc}$ [37] given by the following:

$$\begin{aligned} C &::= A \mid \forall f.C \\ D &::= C \mid \neg C \mid D_1 \sqcap D_2 \mid \forall f.D \mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf} \\ E &::= C \mid \bot \mid E_1 \sqcap E_2 \mid \forall f.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2) \end{aligned}$$

The main idea behind tractability of $\mathcal{CFD}_{nc}$ relies on the fact that left-hand-sides of inclusion dependencies can only observe object membership in a *single atomic concept* (as opposed to a conjunction of concepts). Hence, while models of this logic require exponentially many objects labelled by conjunctions of primitive concepts in general, they can be abstracted in a polynomial way. The construction of the actual model is then similar to the standard NFA to DFA construction followed by unfolding of the resulting DFA.

**Theorem 17 ([37]).** The logical implication problem for $\mathcal{CFD}_{nc}$ is complete for PTIME.

As with the dialect $\mathcal{CFD}$, hardness follows from reducing HornSAT to reasoning with PFDs.

### 4.3 Value Restrictions and Limited Conjunctions

The above has shown that allowing an arbitrary use of concept conjunction on the left-hand-sides of inclusion dependences in a $\mathcal{CFD}_{nc}$ TBox immediately leads to hardness for EXPTIME (a consequence of Theorem 4). The complexity can be traced to the need for exponentially many objects labelled by different sets of primitive concepts to be generated. The following definition provides a way of controlling this need for all such objects:

**Definition 18 (Restricted Conjunction).** Let $k > 0$ be a constant. We say that TBox $\mathcal{T}$ is a $\mathcal{CFD}_{kc}$ TBox if, whenever $\mathcal{T} \models (A_1 \sqcap \cdots \sqcap A_n) \sqsubseteq B$ for some set of primitive concepts $\{A_1, \ldots, A_n\} \cup \{B\}$, with $n > k$, then $\mathcal{T} \models (A_{i_1} \sqcap \cdots \sqcap A_{i_k}) \sqsubseteq B$ for some $k$-sized subset $\{A_{i_1}, \ldots, A_{i_k}\}$ of the primitive concepts $\{A_1, \ldots, A_n\}$. □

A *saturation-style* procedure based on this definition can be implemented to generate *all implied inclusion dependencies* with at most $k$ primitive concepts (value restrictions) on left-hand-sides of inclusion dependencies [26]. The decision procedure essentially follows the procedure for $\mathcal{CFD}_{nc}$ but is exponential in $k$ due

to the need to consider sets of concepts up to size $k$ (essentially by determining all implied inclusion dependencies that are not a trivial weakening of other inclusion dependencies) and leads to the following:

**Theorem 19 ([26]).** The logical implication problem for $\mathcal{CFD}_{kc}$ is complete for PTIME for a fixed value of $k$; the decision procedure is exponential in $k$.

In addition, the procedure enables an incremental means of determining the minimum $k$ for which a given TBox is a $\mathcal{CFD}_{kc}$ TBox, that is, allows for testing if a given parameter $k$ suffices:

**Theorem 20 (Testing for $k$ [26]).** A TBox $\mathcal{T}$ is *not* a $\mathcal{CFD}_{kc}$ TBox if and only if there is an additional single-step inference that infers a non-trivial inclusion dependency (i.e., one that is not a weakening of an already discovered dependency) with $k+1$ conjuncts on the left hand side.

An algorithm based on iterative deepening allows one to determine the value of $k$ for a given TBox in a *pay as you go* way. Hence the decision procedure also runs within the optimal time bound, exponential in $k$ and polynomial in $|\mathcal{T}| + |\mathcal{Q}|$, even when $k$ is *not* part of the input.

### 4.4 Adding Inverse Features

Recall from Sect. 3.3 that we consider only the (unqualified) inverse feature constructor, $\exists f^{-1}$, to be added to the $D$ grammar rules of $\mathcal{CFD}_{nc}$ and $\mathcal{CFD}_{kc}$, yielding the respective logics $\mathcal{CFDI}_{nc}$ and $\mathcal{CFDI}_{kc}$. However, additional restrictions are still required to guarantee tractability of logical consequence [38]. We introduce the restrictions by examples:

1. *Inverses and Value Restrictions.* Interactions between these two concept constructors can be illustrated by the following inference:

$$\{A \sqsubseteq \exists f^{-1}, \forall f.A' \sqsubseteq \forall f.B\} \models A \sqcap A' \sqsubseteq B.$$

   This cannot be allowed since unrestricted use of this construction yields hardness for EXPTIME (see Theorem 4). $\mathcal{CFDI}_{nc}$ syntactically restricts TBoxes to avoid the above situation by requiring additional inclusion dependencies of the form $A \sqsubseteq A'$, $A' \sqsubseteq A$, or $A \sqcap A' \sqsubseteq \bot$ to be present in a TBox whenever the above pattern appears. Note that $\mathcal{CFDI}_{kc}$ does not require this restriction since the *testing for k* procedure we have outlined will detect the above situation (thus determining the *price*).
2. *Inverses and PFDs.* The second interaction that hinders tractability is between inverses and PFDs. In particular, a logical consequence problem of the form

$$\{A \sqsubseteq \exists f^{-1}, \forall f.A \sqsubseteq A, \ldots\} \models (\forall h_1.A) \sqcap (\forall h_2.A) \sqcap (h_1.f = h_2.f) \sqsubseteq h_1 = h_2$$

   will force two infinite $f$ anti-chains starting from two $A$ objects created by the left-hand-side of the posed question. We have shown how to use these

anti-chains and additional PFDs in the TBox to reduce *linearly bounded DTM acceptance* [19] to logical implication in this case, yielding PSPACE-hardness, and how to repair this by further limiting the syntax of PFDs in a way that disables this kind of interaction with inverse features [38]. In particular, PFDs in a TBox must now have one of the following two forms:

- $C : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.\, \mathsf{Pf}_i, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$; and
- $C : \mathsf{Pf}_1, \ldots, \mathsf{Pf} \,.g, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \,.f$, for some primitive features $f$ and $g$.

Inverses obeying these two restrictions can then be added to both the FunDL dialects $\mathcal{CFDI}_{nc}$ and $\mathcal{CFDI}_{kc}$ while maintaining tractability:

**Theorem 21 ([38]).** The logical implication problems for $\mathcal{CFDI}_{nc}$ and $\mathcal{CFDI}_{kc}$ are complete for PTIME, in the latter case for a fixed value of $k$.

## 5 Partial Features, Roles, ABoxes and Query Answering

The third part of our survey considers how partial features and role hierarchies can be accommodated in FunDL dialects, and how to check for knowledge base consistency and to evaluate queries over FunDL knowledge bases consisting of a so-called ABox in addition to a TBox.

### 5.1 Partial Features

We first consider the impact of changing the semantics of features in the FunDL family to *partial features* [25,26,40,41]. The changes can be summarized as follows:

1. Features $f \in \mathsf{F}$ are now interpreted as *partial* functions on $\triangle$ (i.e., the result can be *undefined* for some elements of $\triangle$);
2. A path function $\mathsf{Pf}$ now denotes a partial function resulting from the composition of partial functions;
3. The syntax of $C$ in feature-based DLs is extended with an additional concept constructor, $\exists f$, called an *existential restriction* that can now appear on both sides of inclusion dependencies;
4. The $\exists f$ concept constructor is interpreted as $\{x \mid \exists y \in \triangle.(f)^{\mathcal{I}}(x) = y\}$.
5. We adopt a *strict* interpretation of set membership and equality. This means that set membership holds only when the value exists; and equality holds only when both sides are defined and denote the same object.

In the light of these changes, we need to consider their impact on concept constructors that involve features or feature paths:

*Value Restrictions.* Our definition of value restriction $\forall f.C$ (see Definition 1) assumes features are total. For partial features, there is now a choice:

1. keeping the original semantics, i.e., objects in the interpretation of $\forall f.C$ *must have a feature $f$ defined* and leading to a $C$ object, or

2. altering the semantics to match $\mathcal{ALC}$-style semantics, i.e., the $f$ value of objects in the interpretation of such a value restriction must be a $C$ object, *if such a value exists*; we denote this variant $\widetilde{\forall} f.C$.

While not equivalent, it is easy to see that many inclusion dependencies can be expressed using either variant of the value restriction, for example

$$A \sqsubseteq \widetilde{\forall} f.B \;\text{ can be expressed as }\; A \sqcap \forall f.\top \sqsubseteq \forall f.B.$$

Note that when the original semantics is used, the existential restriction $\exists f$ is simply a synonym for $\forall f.\top$. Also, since features are still *functional*, the so-called *qualified existential restrictions* of the form $\exists f.C$, with semantics given by $(\exists f.C)^{\mathcal{I}} = \{x \mid \exists y \in \triangle.(f)^{\mathcal{I}}(x) = y \wedge y \in (C)^{\mathcal{I}}\}$, can be simulated by expansion to $\exists f \sqcap \forall f.C$. Indeed, hereon we write $\exists \mathsf{Pf}$ as shorthand for $\exists f_1 \sqcap \forall f_1.(\exists f_2 \sqcap \forall f_2.(\ldots(\exists f_k)\ldots))$.

*PFDs.* Our PFDs agree with the definition of identity constraints in [11], where $\mathsf{Pf}_0 = id$, which also require path values to exist. To further clarify the impact of this observation, note that a *PFD inclusion dependency* of the form $C_1 \sqsubseteq C_2 : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0$ is violated when (a) all path functions $\mathsf{Pf}_0, \ldots, \mathsf{Pf}_k$ are defined for a $C_1$ object $e_1$ and a $C_2$ object $e_2$, and (b) $(\mathsf{Pf}_i)^{\mathcal{I}}(e_1) = (\mathsf{Pf}_i)^{\mathcal{I}}(e_2)$ holds only for $1 \leq i \leq k$. Formally, and more explicitly, this leads to the following interpretation of PFDs in the presence of partial features:

$$\begin{array}{l}((C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0))^{\mathcal{I}} = \\ \{x \mid \forall y.y \in (C)^{\mathcal{I}} \wedge x \in ((\exists \mathsf{Pf}_0))^{\mathcal{I}} \wedge y \in ((\exists \mathsf{Pf}_0))^{\mathcal{I}} \wedge \\ \quad \bigwedge_{i=1}^{k}(x \in ((\exists \mathsf{Pf}_i))^{\mathcal{I}} \wedge y \in ((\exists \mathsf{Pf}_i))^{\mathcal{I}} \wedge (\mathsf{Pf}_i)^{\mathcal{I}}(x) = (\mathsf{Pf}_i)^{\mathcal{I}}(y)) \\ \qquad\qquad\qquad\qquad \rightarrow (\mathsf{Pf}_0)^{\mathcal{I}}(x) = (\mathsf{Pf}_0)^{\mathcal{I}}(y)\,\}.\end{array}$$

*Equational Concepts.* Similarly to PFDs, we assume the strict interpretation of equalities, i.e., an object belongs to $(\mathsf{Pf}_1 = \mathsf{Pf}_2)$ if and only if both $\mathsf{Pf}_1$ and $\mathsf{Pf}_2$ are defined for the object and agree.

**Partiality in Expressive FunDL.** In expressive FunDL dialects, partiality can be simulated by introducing an auxiliary primitive concept $G$ that will stand for the *domain of existing objects*. Depending on our choice of semantics for value restrictions we get a mapping of a TBox under the partial semantics to a TBox under the total semantics. We first define a way to *modify concept descriptions* to capture the desired semantics of partiality:

1. $\mathsf{PtoT}(C) = C[\forall f.C \mapsto \neg G \sqcup \forall f.(C \sqcap G)$ for $f \in F]$, for the original semantics,
2. $\mathsf{PtoT}(C) = C[\exists f \mapsto \neg G \sqcup \forall f.G,\ \text{for } f \in F]$ for the $\mathcal{ALC}$-style semantics.

Now we can define a partial to total TBox mapping

$$\mathcal{T}_{\text{total}} = \{G \sqsubseteq \mathsf{PtoT}(C) \mid \top \sqsubseteq C \in \mathcal{T}_{\text{partial}}\} \cup \{\forall f.G \sqsubseteq G \mid f \in F\},$$

and show:

**Theorem 22** ([41])**.** Let $\mathcal{T}_{\text{partial}}$ be a *partial-$\mathcal{DLFI}$* TBox in which all inclusion dependencies are of the form $\top \sqsubseteq C$ with $C$ in negation normal form. Then

$$\mathcal{T}_{\text{partial}} \models \top \sqsubseteq C \iff \mathcal{T}_{\text{total}} \models G \sqsubseteq \mathsf{PtoT}(C),$$

for $G$ a fresh primitive concept.

To extend this construction to the full *partial-$\mathcal{DLFDI}$* logic, it is sufficient to *encode* the path function existence preconditions of PFDs in terms of the auxiliary concept $G$ as follows: if $A \sqsubseteq B : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0 \in \mathcal{T}_{\text{partial}}$ then

$$A \sqcap (\prod_{i=0}^{k} \forall\, \mathsf{Pf}_i\, .G) \sqsubseteq B \sqcap (\prod_{i=0}^{k} \forall\, \mathsf{Pf}_i\, .G) : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0 \qquad (1)$$

is added to $\mathcal{T}_{\text{total}}$. Here, we are assuming w.l.o.g. that $A$ and $B$ are primitive concept names ($\mathcal{DLFD}$ allows one to give such names to complex concepts).

**Theorem 23** ([41])**.** Let $\mathcal{T}_{\text{partial}}$ be a *partial-$\mathcal{DLFDI}$* TBox in which all inclusion dependencies are of the form $\top \sqsubseteq C$ or $A \sqsubseteq B : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0$. Then

$$\begin{array}{l} \mathcal{T}_{\text{partial}} \models \top \sqsubseteq C \iff \mathcal{T}_{\text{total}} \models G \sqsubseteq \mathsf{PtoT}(C), \text{ and} \\ \mathcal{T}_{\text{partial}} \models A \sqsubseteq B : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf} \iff \mathcal{T}_{\text{total}} \models (1), \end{array}$$

for $G$ a fresh primitive concept.

This result can also be extended to the logic $\mathcal{DLFDE}^-$ by appropriately transforming the posed question with respect to the strict interpretation of equational constraints.

**Partiality in Tractable FunDL.** A similar construction can be used to accommodate partial features in tractable FunDL dialects. However, there is a need to accommodate the various restrictions in these logics that guarantee tractability. Hence, we assume that we will be given a *partial-$\mathcal{CFDI}_{kc}$* TBox $\mathcal{T}_{\text{partial}}$ in a normal form, and that the semantics of value restrictions is the same in both the partial and the total logic. We then derive a $\mathcal{CFDI}_{(k+1)c}$ TBox $\mathcal{T}_{\text{total}}$ by applying the following rules:

$$\begin{array}{lrcl} 1. & A \sqsubseteq \bot & \mapsto & A \sqcap G \sqsubseteq \bot \\ 2. & A \sqsubseteq B & \mapsto & A \sqcap G \sqsubseteq B \\ 3. & A \sqcap B \sqsubseteq C & \mapsto & A \sqcap B \sqcap G \sqsubseteq C \\ 4. & A \sqsubseteq \forall f.B & \mapsto & A \sqcap G \sqsubseteq \forall f.B \sqcap \forall f.G \\ 5. & \forall f.A \sqsubseteq B & \mapsto & \forall f.A \sqcap \forall f.G \sqsubseteq B \\ 6. & A \sqsubseteq \exists f & \mapsto & A \sqcap G \sqsubseteq \forall f.G \\ 7. & \exists f \sqsubseteq A & \mapsto & \forall f.G \sqsubseteq A \end{array}$$

and by adding the inclusion dependency $\forall f.G \sqsubseteq G$ to $\mathcal{T}_{\text{total}}$ for each feature.

Conversely, value restrictions in more traditional role-based description logics, such as $\mathcal{ALC}$, also cover the *vacuous cases*, containing objects for which $f$ is

undefined (in addition to the above). This definition unfortunately leads to computational difficulties: the *disjunctive* nature of such a value restriction, when used on left-hand-sides of inclusion dependencies, destroys the canonical model property of the logic. This leads to intractability of query answering as shown by Calvanese *et al.* [12]. To regain tractability, it becomes necessary to restrict the use of value restrictions on the left-hand-side of inclusion dependencies. In a normal form, the $C$ grammar for left-hand-side concepts must replace $\forall f.A$ with $\forall f.A \sqcap \exists f$. This leads to alternative rules when simulating the partial-feature logic in the total-feature counterpart, i.e.,

$$\begin{array}{lrcl} 4'. & A \sqsubseteq \forall f.B & \mapsto & A \sqcap G \sqsubseteq \forall f.B \\ 5'. & (\forall f.A \sqcap \exists f) \sqsubseteq B & \mapsto & (\forall f.A \sqcap \forall f.G) \sqsubseteq B \end{array}$$

The technique for treating posed questions [40] extends to *partial-*$\mathcal{CFDI}_{kc}$ and yields the following:

**Theorem 24 ([40]).** Let $\mathcal{T}_{\text{partial}}$ be a *partial-*$\mathcal{CFDI}_{kc}$ TBox, $\mathcal{Q}_{\text{partial}}$ a posed question, and $\mathcal{T}_{\text{total}}$ be defined as above. Then $\mathcal{T}_{\text{total}}$ is a $\mathcal{CFDI}_{(k+1)c}$ TBox and

$$\mathcal{T}_{\text{partial}} \models \mathcal{Q}_{\text{partial}} \iff \mathcal{T}_{\text{total}} \models \mathcal{Q}_{\text{total}},$$

where $\mathcal{Q}_{\text{total}}$ is effectively constructed from $\mathcal{Q}_{\text{partial}}$ by adding appropriate conjunctions with $G$ concepts.

Since $|\mathcal{Q}_{\text{partial}}|$ is linear in $|\mathcal{Q}_{\text{total}}|$, this provides a tractable decision procedure for logical implication in *partial-*$\mathcal{CFDI}_{kc}$. An analogous result involving *partial-*$\mathcal{CFDI}_{kc}$ knowledge base reasoning was studied in [26].

### 5.2 Simulating Roles and Role Constructors

It is well known that unrestricted use of *role functionality* with *role hierarchies*, e.g., DL-Lite$_{\text{core}}^{\mathcal{HF}}$, leads to intractability [2,10]. Conversely, the ability to reify roles would seem to enable capturing a limited variety of *role hierarchies*.[2]

Consider roles $R$ and $S$ and the corresponding primitive concepts $C_R$ and $C_S$, respectively, and assume that the domains and ranges of the reified roles are captured by the features *dom* and *ran* common to both the reified roles. Subsumption and disjointness of these roles can then be captured as follows:

$$\begin{array}{lcl} R \sqsubseteq S & \mapsto & C_R \sqsubseteq C_S, C_R \sqsubseteq C_S : dom, ran \rightarrow id \quad \text{and} \\ R \sqcap S \sqsubseteq \bot & \mapsto & C_R \sqsubseteq \neg C_S, C_R \sqsubseteq C_S : dom, ran \rightarrow id, \end{array}$$

assuming that the reified role $R$ (and analogously $S$) also satisfies the key constraint $C_R \sqsubseteq C_R : dom, ran \rightarrow id$. Such a reduction does *not* lend itself to capturing role hierarchies between roles and *inverses* of roles (due to fixing the names of the features *dom* and *ran*).

[2] Unlike DL-Lite$_{\text{core}}^{(\mathcal{HF})}$, that restricts the applicability of functional constraints in the presence of role hierarchies, we review what forms of role hierarchies can be captured while retaining the ability to specify arbitrary keys and functional dependencies.

Moreover, for tractable fragments of FunDL, a condition introduced earlier, governing the interactions between inverse features and value restrictions, introduces additional interactions that interfere with (simulating) role hierarchies, in particular in cases when *mandatory participation* constraints are present. Consider again roles $R_1$ and $R_2$ and the corresponding primitive concepts $C_{R_1}$ and $C_{R_2}$, respectively, and associated constraints that declare typing for the roles,

$$C_{R_1} \sqsubseteq \forall dom.A_1, C_{R_1} \sqsubseteq \forall ran.B_1, C_{R_1} \sqsubseteq C_{R_1} : dom, ran \rightarrow id$$
$$C_{R_2} \sqsubseteq \forall dom.A_2, C_{R_2} \sqsubseteq \forall ran.B_2, C_{R_2} \sqsubseteq C_{R_2} : dom, ran \rightarrow id$$

originating, e.g., from an ER diagram postulating that entity sets $A_i$ and $B_i$ participate in a relationship $R_i$ (for $i = 1, 2$). Now consider a situation where the participation of $A_i$ in $R_i$ is *mandatory* (expressed, e.g., as $A_i \sqsubseteq \exists R_i$ in DL-Lite). This leads to the following constraints:

$$A_1 \sqsubseteq \exists dom^{-1}, \forall dom.A_1 \sqsubseteq C_{R_1} \text{ and } A_2 \sqsubseteq \exists dom^{-1}, \forall dom.A_2 \sqsubseteq C_{R_2}.$$

The earlier condition governing the use of inverse roles then requires that one of

$$A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_1, \text{ or } A_1 \sqsubseteq \neg A_2$$

are present in the TBox. The first (and second) conditions imply that $C_{R_1} \sqsubseteq C_{R_2}$ ($C_{R_2} \sqsubseteq C_{R_1}$, respectively). The third condition states that the domains of (the reified versions of) $R_1$ and $R_2$ are disjoint, hence the roles themselves must also be disjoint. Hence, in the presence of $C_{R_1} \sqsubseteq C_{R_2} : dom, ran \rightarrow id$, the concepts $C_{R_1}$ and $C_{R_2}$ must also be disjoint.

All this shows that some form of role hierarchies can be accommodated in FunDL dialects. However:

1. only primitive roles can be captured (i.e., capturing inverse roles will not be possible), and
2. when tractability is required, only *role forests* can be captured, that is, for each pair of roles participating in the same role hierarchy, one must be a super-role of the other or their domain and range features must be distinct.

The first restriction originates in the way (binary) roles are reified—by assigning canonically-named features. This prevents modelling constraints such as $R \sqsubseteq R^-$ (which would seem to require simple equational constraints for feature renaming). The second condition is essential to maintaining tractability of reasoning [38]. Note, however, that no such restriction is needed for roles that do *not* participate in the same role hierarchy; this is achieved by appropriate choice of names for the features *dom* and *ran*.

Last, our approach to role hierarchies can easily be extended to handling hierarchies of higher-arity non-homogeneous relationships (again, via reification and appropriate naming of features) that originate, e.g., from relating the aggregation constructs via inheritance in the EER model [27,28]. The reification based approach differs from approaches to modelling higher arity relationships directly

in the underlying description logic, such as $\mathcal{DLR}$ [13,14] in which only homogeneous relationships can be related in hierarchies. This is due to the positional nature of referring to components of such relationship in lieu of using arguably more flexible *keywords* (realized by features in FunDL).

### 5.3 ABoxes, Knowledge Bases, and Consistency

First we consider the issue of *knowledge bases*, combinations of terminological knowledge (TBoxes) with factual assertions about particular objects (ABoxes).

**Definition 25 (ABoxes and Knowledge Bases).** A knowledge base $\mathcal{K}$ is defined by a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$ consisting of a finite set of facts in form of *concept assertions* $A(a)$, *basic function assertions* $f(a) = b$ and *path function assertions* $\mathsf{Pf}_1(a) = \mathsf{Pf}_2(b)$. $\mathcal{A}$ is called a *primitive* ABox if it consists only of concept and basic function assertions. Semantics is extended to interpret individuals $a$ to be elements of $\triangle$. An interpretation $\mathcal{I}$ satisfies a concept assertion $A(a)$ if $(a)^{\mathcal{I}} \in (A)^{\mathcal{I}}$, 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 $\mathcal{K}$, $(a)^{\mathcal{I}} \neq (b)^{\mathcal{I}}$. □

A standard reasoning problem for knowledge bases is the consistency problem, the question whether a knowledge base has a model. We relate this problem to the logical implication problems for FunDL dialects that admit equational constructs in the posed questions. It turns out that either capacity alone is sufficient: each is able to effectively simulate the other [21].

**ABoxes vs. Equalities in Posed Questions.** Intuitively, path equations can *enforce* that an arbitrary finite graph (with feature-labeled edges and concept description-labeled nodes) is a part of any model that satisfies the equations. Such a graph can equivalently be enforced by an ABox. Hence we have:

**Theorem 26 ([21]).** Let $\mathcal{T}$ be a $\mathcal{DLFD}$ terminology and $\mathcal{A}$ an ABox. Then there is a concept $E$ such that $\mathcal{T} \cup \mathcal{A}$ is not consistent if and only if $\mathcal{T} \models E \sqsubseteq \bot$.

Conversely, it is also possible to show that ABox reasoning can be used for reasoning about equational constraints in the posed questions. However, as the equational concepts are closed under Boolean constructors, a single equational problem may need to map to several ABox consistency problems.

**Theorem 27 ([21]).** Let $\mathcal{T}$ be a $\mathcal{DLFD}$ terminology and $E$ an equational concept. Then there is a finite set of ABoxes $\{\mathcal{A}_i : 0 < i \leq k\}$ such that

$$\mathcal{T} \models E \sqsubseteq \bot \text{ iff } \mathcal{T} \cup \mathcal{A}_i \text{ is not consistent for all } 0 < i \leq k.$$

Theorems 26 and 27 hold even when the terminology $\mathcal{T}$ is a $\mathcal{DLF}$ TBox (i.e., does not contain any occurrences of the PFD concept constructor) or to the tractable FunDL dialects $\mathcal{CFD}$ and $\mathcal{CFDI}_{kc}$. Here, posed question $E$ concepts must be limited to retain a PTIME upper bound in the size of the posed question (Sect. 4 has the details).

### 5.4 Query Answering

Conjunctive queries (CQ) are, as usual, formed from atomic queries (or *atoms*) of the form $C(x)$ and $x.\,\mathsf{Pf}_1 = y.\,\mathsf{Pf}_2$, where $x$ and $y$ are variables, using conjunction and existential quantification. To simplify notation, we conflate conjunctive queries with the set of its constituent atoms and a set of *answer variables*. Given a knowledge base (KB) consisting of a TBox and ABox expressed in terms of a tractable FunDL dialect, our goal is to compute the so called *certain answers*:

**Definition 28 (Certain Answer).** Let $\mathcal{K}$ be a KB over a tractable FunDL dialect and $Q = \{\bar{x} \mid \varphi\}$ a CQ. A *certain answer* to $Q$ over $\mathcal{K}$ is a substitution of constant symbols $\bar{a}$, $[\bar{x} \mapsto \bar{a}]$, such that $\mathcal{K} \models Q[\bar{x} \mapsto \bar{a}]$. □

Computing certain answers in this case requires a combination of *perfect rewriting* [10] and of the *combined approach* [22,36]. The latter is necessary because tractable FunDL dialects are complete for PTIME and first-order rewriting alone followed by evaluating the rewritten query over the ABox will not suffice. The former is necessary to avoid the need for exponentially many anonymous objects in an ABox completion (unlike $\mathcal{EL}$ logics in which there is a need for only polynomially many such objects).

This approach was introduced for $\mathcal{CFDI}_{nc}$ in [17,18] and the two steps are realized by two procedures:

1. $\mathsf{Completion}_{\mathcal{T}}(\mathcal{A})$: this procedure applies consequences of the TBox $\mathcal{T}$ to the ABox $\mathcal{A}$. In particular, concept membership is fully determined for all all ABox individuals. For example, if $\{A(a), f(a) = b, f(b) = c, \ldots\} \subseteq \mathcal{A}$ and $\mathcal{T} \models A \sqsubseteq \forall f.A$, we require $\{A(b), A(c), \ldots\} \subseteq \mathsf{Completion}_{\mathcal{T}}(\mathcal{A})$. (Indeed, propagating concepts along paths that exists in an ABox is the reason why perfect rewriting alone will not suffice in tractable FunDL dialects.)
2. $\mathsf{Fold}_{\mathcal{T}}(Q)$: this procedure rewrites an input CQ to an union of CQs that account for the constraints in $\mathcal{T}$ that postulate existence of anonymous objects in all models of the knowledge base. A (slight simplification of a) typical rule applied during such a rewriting looks as follows:

   If $\{y.f = x, A(y)\} \subseteq \psi$ and $y$ does not appear elsewhere in $\psi$ nor is an answer variable, then $\mathsf{Fold}(Q) := \mathsf{Fold}(Q) \cup \{\{\bar{y} \mid \psi_i\}\}$ for all $\psi_i = \psi - \{y.f = x, A(y)\} \cup \{B_i(x)\}$, where $B_i$ are all maximal primitive concepts w.r.t. $\sqsubseteq$ satisfying the logical implication conditions $\mathcal{T} \models B_i \sqsubseteq \exists f^{-1}$ and $\mathcal{T} \models \forall f.B_i \sqsubseteq A\}$.

   The rule states that whenever the variable $y$ is connected to the rest of the query via a single feature $f$, it may be mapped to an anonymous individual. This is accommodated by the query $\psi_i$ that no longer uses the variable $y$,

but implies $\psi$ since the existence of the necessary individual is implied by the TBox $\mathcal{T}$ and the $B_i(x)$ atom in $\psi_i$.

Note that query rewriting requires a *completed* ABox. Thus, the rewriting produces fewer disjuncts since only maximal concepts need to be retained.

**Theorem 29 ([40]).** Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{CFDI}_{nc}$ knowledge base and $Q$ a conjunctive query. Then

$$\mathcal{K} \models Q[\bar{x} \mapsto \bar{a}] \iff (\emptyset, \mathsf{Completion}_{\mathcal{T}}(\mathcal{A})) \models \mathsf{Fold}_{\mathcal{T}}(Q)[\bar{x} \mapsto \bar{a}].$$

Note that $(\emptyset, \mathsf{Completion}_{\mathcal{T}}(\mathcal{A})) \models \mathsf{Fold}_{\mathcal{T}}(Q)[\bar{x} \mapsto \bar{a}]$ reduces to evaluating the query $\mathsf{Fold}_{\mathcal{T}}(Q)$ over a finite relational structure $\mathsf{Completion}_{\mathcal{T}}(\mathcal{A})$. Tractability (in $|\mathcal{K}|$) then follows from $|\mathsf{Completion}_{\mathcal{T}}(\mathcal{A})|$ being polynomial in $|\mathcal{A}|$ and the fact that reasoning in $\mathcal{K}$ is in PTIME. This approach was later extended to other tractable dialects of FunDL including logics with partial features up to and including *partial-*$\mathcal{CFDI}_{kc}$ [26].

## 6 Related Work

Recall that $\mathcal{CFDI}_{nc}$ is a tractable FunDL dialect in which left-hand-sides of inclusion dependencies exclude the use of negation as well as conjunction. The possibility of the Krom extension of this dialect, that readmits negation, has also been explored [39]. Tractability is still possible, but requires TBoxes to be free of non-key PFDs, requires ABoxes to be primitive, and requires the adoption of UNA. (Relaxing any of these conditions leads to intractability.)

We have also considered how concepts in FunDL dialects can replace constants in an ABox as a way of referring to entities or objects. Indeed, the judicious adoption of features instead of roles in these dialects makes it easy for an ABox to be a window on factual data in backend object-relational data sources. Coupled with the notion of *referring expression types*, this overall development pays off nicely in ontology-based data access and in relating conceptual and object-relational database design in information systems [6,7].

A short review of ways in which PFDs themselves have been generalized completes our survey.

**Path Order Dependencies.** PFDs can be viewed as a variety of tuple generating dependencies in which equality is the only predicate occurring on the right-hand-side. The possibility that any comparison operator can be used instead has also been investigated. In particular, so-called *guarded order dependencies* can be added to the expressive FunDL dialect $\mathcal{DLF}$ without impacting the complexity of logical implication [29]. For our introductory university TBox, a correlation between *gpa* and *mark* can be expressed by such a dependency:

$$\text{TAKES} \sqsubseteq \text{TAKES} : \mathit{class}^{=}, \mathit{mark}^{<} \rightarrow \mathit{student.gpa}^{\leq}$$

The dependency asserts that *the grade point average of a student is never greater than that of another student when there is some class they have both taken in which the latter student obtained a better grade.*

**Regular Path Functional Dependencies.** Left and right-hand-sides of PFDs can be viewed as instances of finite regular languages. The possibility of allowing these languages to be defined by regular expressions admitting the Kleene closure operator has also been investigated. In particular, *regular path functional dependencies* were introduced in [30], and more general *regular path order dependencies* in [33], and, in both cases, were shown to not impact the complexity of logical implication when added to $\mathcal{DLF}$. This remains the case when value restrictions are also generalized by allowing component path expressions to be given by regular expressions. For example, to ensure that *every professor eventually reports to a dean*, one can now add the inclusion dependencies

$$\text{DEAN} \sqsubseteq \text{CHAIR} \quad \text{and} \quad \text{PROF} \sqsubseteq \neg\forall \mathit{reports} * .\neg\text{DEAN}$$

to the university TBox.

**Temporal Path Functional Dependencies.** Finally, adding both a temporal variety of PFDs and a *global model operator* ($\Box$) to $\mathcal{DLF}$ is also possible without impact on the complexity of logical implication [34]. This enables adding the inclusion dependency

$$\text{PERSON} \sqsubseteq (\Box_{\text{forever}}\text{PERSON}) \sqcap (\text{PERSON} : \mathit{id} \rightarrow_{\text{forever}} \mathit{name})$$

to the university TBox to ensure that *a person is always a person* and that *the name of a person never changes*. Adding the inclusion dependency

$$\text{DEPT} \sqsubseteq (\text{DEPT} : \mathit{id} \rightarrow_{\text{term}} \mathit{head}) \sqcap (\text{DEPT} : \mathit{head} \rightarrow_{\text{term}} \mathit{id})$$

would ensure that *a professor is the unique head of a department for a fixed term*. However, it *not* possible to add any form of eventuality together with temporal PFDs to $\mathcal{DLF}$ (e.g., by also adding regular PFDs) and at the same time retain EXPTIME complexity of logical implication for $\mathcal{DLF}$ iteself [34].

## References

1. Ackermann, W.: Uber die Erfullbarkeit gewisser Zahlausdrucke. Math. Ann. **100**, 638–649 (1928)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009). https://doi.org/10.1613/jair.282
3. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005, Edinburgh, Scotland, UK, 30 July–5 August 2005, pp. 364–369. Professional Book Center (2005). http://ijcai.org/Proceedings/05/Papers/0372.pdf
4. Berger, R.: The undecidability of the dominoe problem. Mem. Amer. Math. Soc. **66**, 1–72 (1966)
5. van Emde Boas, P.: The convenience of tilings. In: Complexity, Logic, and Recursion Theory. pp. 331–363. Marcel Dekker Inc. (1997)

6. Borgida, A., Toman, D., Weddell, G.: On referring expressions in information systems derived from conceptual modelling. In: Comyn-Wattiau, I., Tanaka, K., Song, I.-Y., Yamamoto, S., Saeki, M. (eds.) ER 2016. LNCS, vol. 9974, pp. 183–197. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46397-1_14
7. Borgida, A., Toman, D., Weddell, G.E.: On referring expressions in query answering over first order knowledge bases. In: Baral, C., Delgrande, J.P., Wolter, F. (eds.) Proceedings of the Fifteenth International Conference, Principles of Knowledge Representation and Reasoning KR 2016, Cape Town, South Africa, 25–29 April 2016, pp. 319–328. AAAI Press (2016). http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12860
8. Borgida, A., Weddell, G.: Adding uniqueness constraints to description logics. In: Bry, F., Ramakrishnan, R., Ramamohanarao, K. (eds.) DOOD 1997. LNCS, vol. 1341, pp. 85–102. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63792-3_10
9. Brewka, G., Lang, J. (eds.): Principles of knowledge representation and reasoning. In: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, 16–19 September 2008. AAAI Press (2008)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J. Autom. Reasoning **39**(3), 385–429 (2007). https://doi.org/10.1007/s10817-007-9078-x
11. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Path-based identification constraints in description logics. In: Brewka and Lang [9], pp. 231–241. http://www.aaai.org/Library/KR/2008/kr08-023.php
12. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artif. Intell. **195**, 335–360 (2013). https://doi.org/10.1016/j.artint.2012.10.003
13. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Mendelzon, A.O., Paredaens, J. (eds.) Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1–3 June 1998, Seattle, Washington, USA. pp. 149–158. ACM Press (1998). https://doi.org/10.1145/275487.275504
14. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: Nebel, B. (ed.) Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, 4–10 August 2001. pp. 155–160. Morgan Kaufmann (2001). http://ijcai.org/proceedings/2001-1
15. Chomicki, J., Imielinski, T.: Finite representation of infinite query answers. ACM Trans. Database Syst. **18**(2), 181–223 (1993). https://doi.org/10.1145/151634.151635
16. Fürer, M.: Alternation and the ackermann case of the decision problem. L'Enseignement Math. **27**, 137–162 (1981)
17. Jacques, J.S., Toman, D., Weddell, G.E.: Object-relational queries over $\text{cfdi}_{\text{nc}}$ knowledge bases: OBDA for the SQL-Literate. In: Kambhampati, S. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, pp. 1258–1264. IJCAI/AAAI Press (2016). http://www.ijcai.org/Abstract/16/182

18. Jacques, J.S., Toman, D., Weddell, G.E.: Object-relational queries over cfdi_nc knowledge bases: OBDA for the SQL-Literate (extended abstract). In: Lenzerini, M., Peñaloza, R. (eds.) Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, 22–25 April 2016. CEUR Workshop Proceedings, vol. 1577. CEUR-WS.org (2016). http://ceur-ws.org/Vol-1577/paper_10.pdf
19. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations. The IBM Research Symposia Series, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
20. Khizder, V.L., Toman, D., Weddell, G.: Reasoning about duplicate elimination with description logic. In: Lloyd, J., et al. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1017–1032. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44957-4_68
21. Khizder, V.L., Toman, D., Weddell, G.E.: Adding aboxes to a description logic with uniqueness constraints via path agreements. In: Calvanese, D., et al. (eds.) Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8–10 June 2007. CEUR Workshop Proceedings, vol. 250. CEUR-WS.org (2007). http://ceur-ws.org/Vol-250/paper_69.pdf
22. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in DL-Lite. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Proceedings of the Twelfth International Conference Principles of Knowledge Representation and Reasoning, KR 2010, Toronto, Ontario, Canada, 9–13 May 2010. AAAI Press (2010). http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1282
23. Liu, L., Özsu, M.T. (eds.): Encyclopedia of Database Systems. Springer, US (2009). https://doi.org/10.1007/978-0-387-39940-9
24. Machtey, M., Young, P.: An Introduction to the General Theory of Algorithms. North-Holland, Amsterdam (1978)
25. McIntyre, S., Borgida, A., Toman, D., Weddell, G.E.: On limited conjunctions in polynomial feature logics, with applications in OBDA. In: Thielscher, M., Toni, F., Wolter, F. (eds.) Proceedings of the Sixteenth International Conference Principles of Knowledge Representation and Reasoning, KR 2018, Tempe, Arizona, 30 October–2 November 2018, pp. 655–656. AAAI Press (2018). https://aaai.org/ocs/index.php/KR/KR18/paper/view/18016
26. McIntyre, S., Borgida, A., Toman, D., Weddell, G.E.: On limited conjunctions and partial features in parameter tractable feature logics. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, 27 January–1 February 2019, Honolulu, Hawaii, U.S.A. (2019, in press)
27. Song, I., Chen, P.P.: Entity relationship model. In: Liu and Özsu [23], pp. 1003–1009. https://doi.org/10.1007/978-0-387-39940-9_148
28. Thalheim, B.: Extended entity-relationship model. In: Liu and Özsu [23], pp. 1083–1091. https://doi.org/10.1007/978-0-387-39940-9_157
29. Toman, D., Weddell, G.E.: On attributes, roles, and dependencies in description logics and the ackermann case of the decision problem. In: Goble, C.A., McGuinness, D.L., Möller, R., Patel-Schneider, P.F. (eds.) Proceedings of the Working Notes of the 2001 International Description Logics Workshop (DL-2001), Stanford, CA, USA, 1–3 August 2001. CEUR Workshop Proceedings, vol. 49. CEUR-WS.org (2001). http://ceur-ws.org/Vol-49/TomanWeddell-76start.ps

30. Toman, D., Weddell, G.E.: On reasoning about structural equality in XML: a description logic approach. Theor. Comput. Sci. **336**(1), 181–203 (2005). https://doi.org/10.1016/j.tcs.2004.10.036
31. Toman, D., Weddell, G.E.: On the interaction between inverse features and path-functional dependencies in description logics. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005, Edinburgh, Scotland, UK, 30 July–5 August 2005, pp. 603–608. Professional Book Center (2005). http://ijcai.org/Proceedings/05/Papers/1421.pdf
32. Toman, D., Weddell, G.: On keys and functional dependencies as first-class citizens in description logics. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 647–661. Springer, Heidelberg (2006). https://doi.org/10.1007/11814771_52
33. Toman, D., Weddell, G.: On order dependencies for the semantic web. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 293–306. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75563-0_21
34. Toman, D., Weddell, G.E.: Identifying objects over time with description logics. In: Brewka and Lang [9], pp. 724–732. http://www.aaai.org/Library/KR/2008/kr08-071.php
35. Toman, D., Weddell, G.E.: On keys and functional dependencies as first-class citizens in description logics. J. Autom. Reasoning **40**(2–3), 117–132 (2008). https://doi.org/10.1007/s10817-007-9092-z
36. Toman, D., Weddell, G.E.: Applications and extensions of PTIME description logics with functional constraints. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, Pasadena, California, USA, 11–17 July 2009, pp. 948–954 (2009). http://ijcai.org/Proceedings/09/Papers/161.pdf
37. Toman, D., Weddell, G.: Conjunctive query answering in $\mathcal{CFD}_{nc}$: a PTIME description logic with functional constraints and disjointness. In: Cranefield, S., Nayak, A. (eds.) AI 2013. LNCS (LNAI), vol. 8272, pp. 350–361. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03680-9_36
38. Toman, D., Weddell, G.: On adding inverse features to the description logic $\mathcal{CFD}_{nc}^{\forall}$. In: Pham, D.-N., Park, S.-B. (eds.) PRICAI 2014. LNCS (LNAI), vol. 8862, pp. 587–599. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13560-1_47
39. Toman, D., Weddell, G.: On the krom extension of $\mathcal{CFDI}_{nc}^{\forall-}$. In: Pfahringer, B., Renz, J. (eds.) AI 2015. LNCS (LNAI), vol. 9457, pp. 559–571. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26350-2_50
40. Toman, D., Weddell, G.: On partial features in the $\mathcal{DLF}$ Family of Description Logics. In: Booth, R., Zhang, M.-L. (eds.) PRICAI 2016. LNCS (LNAI), vol. 9810, pp. 529–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42911-3_44
41. Toman, D., Weddell, G.E.: On partial features in the DLF dialects of description logic with inverse features. In: Artale, A., Glimm, B., Kontchakov, R. (eds.) Proceedings of the 30th International Workshop on Description Logics, Montpellier, France, 18–21 July 2017. CEUR Workshop Proceedings, vol. 1879. CEUR-WS.org (2017). http://ceur-ws.org/Vol-1879/paper44.pdf