On Partial Features in the $\mathcal{DLF}$ Family of Description Logics

David Toman(✉) and Grant Weddell

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

Abstract. The $\mathcal{DLF}$ family of description logics are fragments of first order logic with underlying signatures based on unary predicate symbols, called atomic concepts, and unary function symbols interpreted as total functions, called features. We show how computational properties relating to a key reasoning service for dialects of this family are preserved when (a) unary function symbols are now interpreted as partial functions, and when (b) a concept constructor is admitted that can characterize circumstances in which partial functions become total.

1 Introduction

The $\mathcal{DLF}$ family of *description logics* (DLs) have been designed primarily to support reasoning tasks about object relational data sources. This includes the $\mathcal{CFD}$ sub-family that admits PTIME algorithms for many of these tasks. Unlike the usual case of *role*-based DLs [1], dialects in this family are *feature*-based, that is, are fragments of first order logic with underlying signatures that replace binary predicate symbols, called roles, with unary function symbols, interpreted as total functions, called features. Since features are intended to capture the notion of an attribute in a data source, a column for a relational table for example, this practice has led to some cognitive incongruity: one must reconcile that *every attribute is fundamentally defined for every object* and introduce protocols for indirectly saying when an attribute is or is not meaningful for various kinds of objects.

In this paper, we show how computational properties of DLs in the $\mathcal{DLF}$ family are preserved when they are modified to address such incongruity. This modification is as follows. First, features, now called *partial features*, are instead interpreted as partial functions, and second, a concept constructor is added that makes it possible to refer to all objects that *have* a value for a given partial feature. The added constructor yields to an ability for any of the DLs to define cases in which partial functions become total functions, such as to say that *every employee has a salary*, or, for two of the DLs considered, to define cases in which partial functions are not meaningful, such as to say that *departments do not have a salary*.

We consider the particular problem of reasoning about *logical implication* for three representative members of the $\mathcal{DLF}$ family: $\mathcal{DLFD}$ [5–7], $\mathcal{CFD}$ [3,8] and $\mathcal{CFD}_{nc}^{\forall}$ [9,10]. The first is a very expressive dialect for which logical implication is

R. Booth and M.-L. Zhang (Eds.): PRICAI 2016, LNAI 9810, pp. 529–542, 2016.
DOI: 10.1007/978-3-319-42911-3_44

EXPTIME-complete, while the remaining two are also members of the $\mathcal{CFD}$ family and therefore have PTIME decision procedures for this problem. Note that all three include a concept constructor for capturing keys and functional dependencies in object relational data sources and can express, for example, that *no two departments have the same manager*, or that *an employee's pay grade determines her salary*. The constructor is called a *path functional dependency* (PFD).

Our contributions, in the order presented, are as follows. Note that, in presenting them and for the remainder of the paper, we write *partial*$-\mathcal{DLFD}$, *partial*$-\mathcal{CFD}$ and *partial*$-\mathcal{CFD}^{\forall}_{nc}$ to refer, respectively, to $\mathcal{DLFD}$, $\mathcal{CFD}$ and $\mathcal{CFD}^{\forall}_{nc}$ when they are presumed to be modified to support partial functions in the above fashion:

1. We introduce a semantics for PFDs when features are partial functions that is entirely neutral on issues of *feature existence*, that is, on whether certain kinds of objects *must have* or even *can have* values for particular features;[1]
2. We show that logical implication for *partial*$-\mathcal{DLFD}$ reduces to logical implication for $\mathcal{DLFD}$, in the process showing that partial functions can be simulated in a straightforward fashion in this dialect; and
3. We show that logical implication for both *partial*$-\mathcal{CFD}$ and *partial*$-\mathcal{CFD}^{\forall}_{nc}$ remains in PTIME by exhibiting refinements of existing respective PTIME decision procedures for deciding logical implication with $\mathcal{CFD}$ and $\mathcal{CFD}^{\forall}_{nc}$.

We conclude with summary comments and an outline of possible directions for future work, with a particular focus on $\mathcal{CFD}^{\forall}_{nc}$ in the latter case: on partial functions for a recent extension, and on relaxing syntactic restrictions to enable straightforward partial function simulation, as we show is possible for $\mathcal{DLFD}$.

2 Background and Definitions

In this section, we begin by reviewing the basic definitions for member dialects of the $\mathcal{DLF}$ family in which features are interpreted as total functions, and then proceed to introduce modifications that yield support for partial features, that is, features that are instead interpreted as partial functions.

Note that DLs in this family do not forgo the ability to capture roles or indeed n-ary relations in general. This can be accomplished with the simple expedient of reification via features, and by using the above-mentioned PFD concept constructor common to these dialects to ensure a set semantics for reified relations. Indeed, the first dialect we consider, $\mathcal{DLFD}$, can capture very expressive role-based dialects, including dialects with so-called qualified number restrictions, inverse roles, role hirarchies, and so on [5].

[1] Such issues can be (and we believe should be) explicitly addressed elsewhere in an ontology.

Definition 1 (Feature-Based DLs). Let F and 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.

Syntax	Semantics: Defn of "$\cdot^{\mathcal{I}}$"	
$C ::= \mathrm{A}$	$\mathrm{A}^{\mathcal{I}} \subseteq \triangle$	(primitive concept; $\mathrm{A} \in \mathsf{PC}$)
$\mid C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(conjunction)
$\mid C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(disjunction)
$\mid \neg C$	$\triangle \setminus C^{\mathcal{I}}$	(negation)
$\mid \forall\, \mathsf{Pf}\,.C$	$\{x : \mathsf{Pf}^{\mathcal{I}}(x) \in C^{\mathcal{I}}\}$	(value restriction)
$\mid C : \mathsf{Pf}_1, ..., \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0$	$\{x : \forall\, y \in C^{\mathcal{I}}. \bigwedge_{i=1}^{k} \mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y) \rightarrow \mathsf{Pf}_0^{\mathcal{I}}(x) = \mathsf{Pf}_0^{\mathcal{I}}(y)\}$	(PFD)
$\mid \top$	$\triangle$	(top)
$\mid \bot$	$\emptyset$	(bottom)
$\mid (\mathsf{Pf}_1 = \mathsf{Pf}_2)$	$\{x : \mathsf{Pf}_1^{\mathcal{I}}(x) = \mathsf{Pf}_2^{\mathcal{I}}(x)\}$	(same-as)

Fig. 1. Syntax and semantics of $\mathcal{DLFD}/\mathcal{CFD}$ concepts.

The *semantics* of expressions is defined with respect to a structure $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $\cdot^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concepts A to be subsets of $\triangle$ and primitive features f to be total functions $f^{\mathcal{I}} : \triangle \rightarrow \triangle$. 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 $\mathcal{T}$. □

In the following, we simplify the notation for path expressions by allowing a syntactic composition $\mathsf{Pf}_1\,.\,\mathsf{Pf}_2$ that stands for their concatenation.

In this presentation, we do *not* consider so-called ABoxes (sets of assertions about membership of individuals in descriptions) and the associated problem of knowledge base consistency. However, these can be reduced to logical implication problems involving posed questions that utilize value restrictions and equational *same-as* descriptions [8].

Also note that the logical implication problem for TBoxes and posed questions characterized so far, that allow arbitrary concepts in inclusion dependencies, is not decidable for a variety of reasons (e.g., see [7] for one case involving

arbitrary PFDs and ABoxes encoded in the above manner). However, restrictions on occurrences of concept constructors has led to a number of decidable fragments that range from light-weight to expressive dialects of feature-based DLs. The restrictions that obtain $\mathcal{DLFD}$, $\mathcal{CFD}$ and $\mathcal{CFD}_{nc}^{\forall}$, the focus of our attention, are given in Sect. 3 for the first case and in Sect. 4 for the remaining two cases.

The two definitions that follow now introduce the necessary modifications to our characterization of feature-based DLs to accommodate *partial features*, that is, features that are interpreted as partial functions. Note that their presentation relies on our notational convention given in our introductory comments of qualifying particular dialects with the word "*partial*" whenever we intend such modifications to apply, as in *partial*$-\mathcal{DLFD}$ for example.

Definition 2 (Partial Features and Existential Restrictions). The syntax of feature-based DLs is extended with an additional concept constructor of the form $\exists f$, called an *existential restriction*. Semantics is updated as follows:

1. features $f \in F$ are now interpreted as *partial* functions on $\triangle$ (i.e., the result can be *undefined* for parts of $\triangle$); and
2. the $\exists f$ concept constructor is interpreted as $\{x : \exists y \in \triangle . f^{\mathcal{I}}(x) = y\}$.

Also, in this setting, path functions (Pf) naturally denote composition of partial functions yielding a partial function, equality ($=$) is true only when both of its arguments are defined (in addition to being equal), set membership ($\in$) requires only defined values to be members of its right hand side argument, etc.[2] □

Observe that features are still *functional*, and that there is therefore no need for a qualified existential restriction of the form $\exists f.C$, with the standard meaning $(\exists f.C)^{\mathcal{I}}$ given by

$$\{x : \exists y \in \triangle . f^{\mathcal{I}}(x) = y \wedge y \in C^{\mathcal{I}}\}.$$

Indeed, they can be simulated using the following identity:

$$\exists f.C = \exists f \sqcap \forall f.C.$$

Using this identity, we write $\exists \mathsf{Pf}$ in the following as shorthand for

$$\exists f_1 \sqcap \forall f_1.(\exists f_2 \sqcap \forall f_2.(\ldots(\exists f_k)\ldots)).$$

On interpreting the PFD constructor in the presence of partial features: the minimum necessary (and we believe most natural) circumstance in which one obtains a violation of a *PFD inclusion dependency* of the form

$$C_1 \sqsubseteq C_2 : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0$$

happens when 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 in which $\mathsf{Pf}_i^{\mathcal{I}}(e_1) = \mathsf{Pf}_i^{\mathcal{I}}(e_2)$ holds only for $i > 0$. This yields the

[2] This arrangement is common and is referred to as the *strict* interpretation of undefined values.

following modification to the interpretation of PFDs in the presence of partial features that we now adopt:

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

Observe that this definition coincides with the original semantics of the PFD constructor given in Fig. 1 when features are interpreted as total functions. Also note that, *without* this modification to semantics, the strict interpretation of undefined values would mean that satisfying the left-hand-side of a PFD would *imply the existence of "*Pf_0 *paths"* (and equality of the "endpoint" of these paths), a circumstance that would violate feature existence neutrality of PFDs mentioned in our introductory comments that seems desirable.

We now return to examples of constraints mentioned in our introductory comments to illustrate the use of existential restrictions and PFDs in $\mathcal{DLF}$ dialects with partial features. Each can be expressed in *partial*$-\mathcal{DLFD}$ and *partial*$-\mathcal{CFD}_{nc}^{\forall}$, and each but the second mentioning negation in *partial*$-\mathcal{CFD}$:

1. EMP $\sqsubseteq \exists$*salary* (*every employee has a salary*);
2. DEPT $\sqsubseteq \neg\exists$*salary* (*departments do not have a salary*);
3. DEPT $\sqsubseteq$ DEPT : *manager* $\rightarrow$ *id* (*no two departments have the same manager*); and
4. EMP $\sqsubseteq$ EMP : *paygrade* $\rightarrow$ *salary* (*employee pay grades determine salaries*).

3 Expressive Feature Logics: The $\mathcal{DLF}$ Family

In this section, we consider the impact of partial features in expressive feature-based description logics, namely in $\mathcal{DLF}$ and $\mathcal{DLFD}$ [5–7]. $\mathcal{DLF}$ allows both TBox and posed question dependencies to contain concepts formed from primitive concepts and bottom using negation, conjunction, disjunction, and restriction concept constructors. $\mathcal{DLFD}$ in addition allows the PFD concept constructor to appear on the right hand sides of inclusion dependencies. These restrictions on syntax yield an expressive Boolean complete description logic with a logical implication problem that is complete for EXPTIME. Additional extensions, e.g., allowing PFDs on the left-hand sides of inclusion dependencies or equational constraints in the posed questions (or equivalently ABoxes) leads to undecidability [7].

We now proceed to demonstrate that partial features can be effectively *simulated* in the original logics by introducing an auxiliary primitive concept G that stands for *existing or generated* objects, and by using *value restrictions* to assign membership of objects generated by the $\exists f$ constructor to this concept. All remaining inclusion dependencies are then simply preconditioned by this auxiliary concept.

Formally, let $\mathcal{T}$ be a *partial*$-\mathcal{DLF}$ TBox in which all inclusion dependencies are of the form $\top \sqsubseteq C$. We define a $\mathcal{DLF}$ TBox $\mathcal{T}_{\mathcal{DLF}}$ as

$$\mathcal{T}_{\mathcal{DLF}} = \{G \sqsubseteq C[\exists f \mapsto \forall f.G, \text{ for all } f \in F] \mid \top \sqsubseteq C \in \mathcal{T}\} \cup \{\forall f.G \sqsubseteq G \mid f \in F\},$$

where G is a primitive concept not occurring in $\mathcal{T}$. Note that the substitution $[\exists f \mapsto \forall f.G, \text{ for all } f \in F]$ is applied simultaneously to *all* occurrences of the $\exists f$ constructor in the concept C.

Theorem 3. Let $\mathcal{T}$ be a *partial*$-\mathcal{DLF}$ TBox in which all inclusion dependencies are of the form $\top \sqsubseteq C$. Then

$$\mathcal{T} \models \top \sqsubseteq C \text{ if and only if } \mathcal{T}_{\mathcal{DLF}} \models G \sqsubseteq C[\exists f \mapsto \forall f.G, \text{ for all } f \in F],$$

for G a fresh primitive concept.

Proof (sketch): For any $\mathcal{I}$ where $\mathcal{I} \models \mathcal{T}_{\mathcal{DLF}}$, we can define an interpretation $\mathcal{J} = (G^{\mathcal{I}}, \cdot^{\mathcal{I}}_{\lceil G^{\mathcal{I}}})$. It is easy to verify that $\mathcal{J} \models \mathcal{T}$ and also that $\mathcal{J} \models \top \sqsubseteq C$ since $\mathcal{I} \models G \sqsubseteq C[\exists f \mapsto \forall f.G, \text{ for all } f \in F]$.

For the other direction, we need to extend a model $\mathcal{J}$ of $\mathcal{T}$ to a model $\mathcal{I}$ of $\mathcal{T}_{\mathcal{DLF}}$ by setting $G^{\mathcal{I}} = \triangle^{\mathcal{J}}$ and by adding *missing* features connecting $\mathcal{I}$ to complete F^* trees with all nodes in $(\neg G)^{\mathcal{I}}$. This way, either $\mathcal{I}$ coincides with $\mathcal{J}$ or satisfies dependencies in $\mathcal{T}_{\mathcal{DLF}}$ and $G \sqsubseteq C[\exists f \mapsto \forall f.G, \text{ for all } f \in F]$ vacuously. □

To extend this construction to the full *partial*$-\mathcal{DLFD}$ logic, it is sufficient to *encode* the path function existence preconditions in terms of the auxiliary concept G as follows: if $A \sqsubseteq B : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}_0 \in \mathcal{T}$ 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 $\mathcal{T}_{\mathcal{DLFD}}$. 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 4. Let $\mathcal{T}$ be a *partial*$-\mathcal{DLFD}$ 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

$\mathcal{T} \models \top \sqsubseteq C$ if and only if $\mathcal{T}_{\mathcal{DLFD}} \models C \sqsubseteq D[\exists f \mapsto \forall f.G, \text{ for all } f \in F]$, and
$\mathcal{T} \models A \sqsubseteq B : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$ if and only if $\mathcal{T}_{\mathcal{DLFD}} \models (1)$,

for G a fresh primitive concept.

Proof (sketch): Logical implication in $\mathcal{DLFD}$ can be reduced to logical implication in $\mathcal{DLF}$ [5–7]. Hence the claim holds by observing that (*) captures properly the semantics of PFDs and then by appealing to Theorem 3. □

Corollary 5. Logical implication is EXPTIME-complete for *partial*$-\mathcal{DLF}$ and for *partial*$-\mathcal{DLFD}$. □

Similar results can be obtained for other members of the $\mathcal{DLF}$ family.

4 Tractable Logics: The $\mathcal{CFD}$ Family

We now consider how partial features impact logical consequence for light-weight (PTIME) feature-based description logics, namely $\mathcal{CFD}$ [3,8] and $\mathcal{CFD}^{\forall}_{nc}$ [9,10]. Both of these logics allow the use of an ABox. Hence, PFDs must adhere to one of the following two forms to avoid undecidability [7]:

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

With this restriction, originally introduced in [3], posed questions can contain inclusion dependencies formed from concepts in Fig. 1 (with a few mild restrictions when tractability in the size of the posed question is required). For simplicity, however, we assume that the concepts in the posed question $\mathcal{Q} = E_1 \sqsubseteq E_2$ adhere to the following grammar:

$$E ::= A \mid \bot \mid E \sqcap E \mid \forall\, \mathsf{Pf} \,.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2).$$

More complex posed questions, e.g., ones that contain the PFD constructor [8], can be equivalently expressed in the above grammar (perhaps as a sequence of posed questions).

Note that, due to syntactic restrictions on TBox inclusion dependencies in $\mathcal{CFD}$ and $\mathcal{CFD}^{\forall}_{nc}$ (see below), we will *not* be able to directly simulate partial features as was done with $\mathcal{DLF}$ and $\mathcal{DLFD}$ above. However, the approach to extending/modifying existing decision procedures for logical implication in the respective logics is analogous: in both cases we introduce an additional unary predicate $\mathsf{D}(x)$ to *mark* the *necessarily existing* objects and use this predicate to restrict the applications of inclusion dependencies. This in turn simulates partial features.

4.1 *partial*$-\mathcal{CFD}$

To obtain a PTIME decision procedure for $\mathcal{CFD}$, we need to further restrict the inclusion dependencies allowed in the TBox $\mathcal{T}$ as follows:

1. left hand sides must be conjunctions of primitive concepts, and
2. right hand sides must be primitive concepts, conjunctions, value restrictions, existential restrictions, and PFDs (obeying restrictions in (2)).

With these restrictions we can show that the logical implication problem for *partial*$-\mathcal{CFD}$ is in PTIME. Our proof is based on encoding a given problem as a collection of Horn clauses. The reduction introduces terms that correspond to path expressions, and relies on the fact that the number of required terms is polynomial in the size of the problem itself.

$$\mathsf{D}(\mathsf{Pf}_1.f) \rightarrow \mathsf{D}(\mathsf{Pf}_1)$$
$$\mathsf{D}(\mathsf{Pf}_1) \rightarrow \mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_1)$$

$$\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2) \rightarrow \mathsf{E}(\mathsf{Pf}_2, \mathsf{Pf}_1)$$
$$\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2) \wedge \mathsf{E}(\mathsf{Pf}_2, \mathsf{Pf}_3) \rightarrow \mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_3)$$
$$\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2) \wedge \mathsf{D}(\mathsf{Pf}_1.f) \wedge \mathsf{D}(\mathsf{Pf}_2.f) \rightarrow \mathsf{E}(\mathsf{Pf}_1.f, \mathsf{Pf}_2.f), \text{ for } \{\mathsf{Pf}_1.f, \mathsf{Pf}_2.f\} \subseteq \mathsf{PF}(\mathcal{T}, \mathcal{Q})$$
$$\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2) \wedge \mathsf{C}_{\mathrm{C}}(\mathsf{Pf}_1) \rightarrow \mathsf{C}_{\mathrm{C}}(\mathsf{Pf}_2)$$

$$\mathsf{C}_{\mathrm{C}_1 \sqcap \mathrm{C}_2}(\mathsf{Pf}) \rightarrow \mathsf{C}_{\mathrm{C}_1}(\mathsf{Pf}) \text{ and } \mathsf{C}_{\mathrm{C}_1 \sqcap \mathrm{C}_2}(\mathsf{Pf}) \rightarrow \mathsf{C}_{\mathrm{C}_2}(\mathsf{Pf})$$
$$\mathsf{C}_{\exists f}(\mathsf{Pf}) \wedge \mathsf{D}(\mathsf{Pf}) \rightarrow \mathsf{D}(\mathsf{Pf}.f) \text{ for all } \mathsf{Pf}.f \in \mathsf{PF}(\mathcal{T}, \mathcal{Q})$$
$$\mathsf{C}_{\forall \mathsf{Pf}'.\mathrm{C}}(\mathsf{Pf}) \rightarrow \mathsf{C}_{\mathrm{C}}(\mathsf{Pf}.\mathsf{Pf}') \text{ for } \mathsf{Pf}.\mathsf{Pf}' \in \mathsf{PF}(\mathcal{T}, \mathcal{Q})$$
$$\mathsf{C}_{(\mathsf{Pf}_1 = \mathsf{Pf}_2)}(\mathsf{Pf}) \wedge \mathsf{D}(\mathsf{Pf}.\mathsf{Pf}_1) \wedge \mathsf{D}(\mathsf{Pf}.\mathsf{Pf}_2) \rightarrow \mathsf{E}(\mathsf{Pf}.\mathsf{Pf}_1, \mathsf{Pf}.\mathsf{Pf}_2)$$
$$\mathsf{C}_{\mathrm{C}:\mathsf{Pf}_1,\ldots,\mathsf{Pf}_k \rightarrow \mathsf{Pf}_0}(\mathsf{Pf}) \wedge \mathsf{C}_{\mathrm{C}}(\mathsf{Pf}') \wedge (\textstyle\bigwedge_{0<i\leq k} \mathsf{E}(\mathsf{Pf}.\mathsf{Pf}_i, \mathsf{Pf}'.\mathsf{Pf}_i)) \wedge \mathsf{D}(\mathsf{Pf}.\mathsf{Pf}_0) \wedge \mathsf{D}(\mathsf{Pf}'.\mathsf{Pf}_0) \rightarrow \mathsf{E}(\mathsf{Pf}.\mathsf{Pf}_0, \mathsf{Pf}'.\mathsf{Pf}_0))$$

$$\mathsf{C}_{\mathrm{A}_1}(\mathsf{Pf}) \wedge \ldots \wedge \mathsf{C}_{\mathrm{A}_k}(\mathsf{Pf}) \rightarrow \mathsf{C}_{\mathrm{D}}(\mathsf{Pf}) \text{ for all } (\mathrm{A}_1 \sqcap \ldots \sqcap \mathrm{A}_k \sqsubseteq \mathrm{D}) \in \mathcal{T}$$
$$\mathsf{C}_{\mathrm{A}}(\mathsf{Pf}.f) \rightarrow \mathsf{C}_{\mathrm{D}}(\mathsf{Pf}) \text{ for all } (\forall f.\mathrm{A} \sqsubseteq \mathrm{D}) \in \mathcal{T}$$

Fig. 2. Expansion rules.

Definition 6 (Expansion Rules). Let $\mathcal{T}$ and $\mathcal{Q}$ be a *partial*$-\mathcal{CFD}$ terminology and a posed question, respectively. We write $\mathsf{CON}(\mathcal{T}, \mathcal{Q})$ to denote the set of all subconcepts appearing in $\mathcal{T}$ and $\mathcal{Q}$, define $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ to be the set

$$\{\mathsf{Pf}.\mathsf{Pf}' \mid \mathsf{Pf} \textit{ is a prefix of a path expression in } \mathcal{Q} \textit{ and } \mathsf{Pf}' \textit{ is a feature occurring in } \mathcal{T} \textit{ or } id\},$$

write C_C to denote unary predicates for $C \in \mathsf{CON}(\mathcal{T}, \mathcal{Q})$, and introduce a unary predicate D and a binary predicate E, with all predicates ranging over the universe $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$. The *expansion rules* for a given terminology $\mathcal{T}$, denoted $\mathsf{R}(\mathcal{T})$, are defined in Fig. 2.[3]

A *goal* for each concept E is a set of ground assertions defined as follows:

$$\mathsf{G}_E = \begin{cases} \{\mathsf{C}_{\mathrm{A}}(id), \mathsf{D}(id)\} & \text{for } E = \mathrm{A}; \\ \{\mathsf{C}_{\bot}(id), \mathsf{D}(id)\} & \text{for } E = \bot; \\ \{\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2), \mathsf{D}(\mathsf{Pf}_1), \mathsf{D}(\mathsf{Pf}_2)\} & \text{for } E = (\mathsf{Pf}_1 = \mathsf{Pf}_2); \\ \mathsf{G}_{E_1} \cup \mathsf{G}_{E_2} & \text{for } E = E_1 \sqcap E_2; \text{ and} \\ \{\mathsf{C}_C(\mathsf{Pf}'.\mathsf{Pf}) \mid \mathsf{C}_C(\mathsf{Pf}) \in \mathsf{G}_{E'}\} & \\ \quad \cup \{\mathsf{D}(\mathsf{Pf}'.\mathsf{Pf}) \mid \mathsf{D}(\mathsf{Pf}) \in \mathsf{G}_{E'}\} & \\ \quad \cup \{\mathsf{E}(\mathsf{Pf}'\,\mathsf{Pf}_1, \mathsf{Pf}'.\mathsf{Pf}_2) \mid \mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2) \in \mathsf{G}_{E'}\} & \text{for } E = \forall\, \mathsf{Pf}'.E'. \end{cases}$$

Given two concept descriptions E_1 and E_2, we say that

$$\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\} \models \mathsf{G}_{E_2}$$

[3] The last rule in the figure does not apply to *partial*$-\mathcal{CFD}$ and is added w.l.o.g. in preparation for treating *partial*$-\mathcal{CFD}^{\forall}_{nc}$. This rule is neither necessary nor applicable in the *partial*$-\mathcal{CFD}$ case.

if $\mathsf{G}_{E_2} \subseteq M$ for every minimal ground model M of $\mathsf{R}(\mathcal{T})$ over $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ that contains $\mathsf{C}_{E_1}(id)$ and $\mathsf{D}(id)$. □

Intuitively, $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ represents a finite graph of objects, predicates $\mathsf{E}(\mathsf{Pf}_1, \mathsf{Pf}_2)$ express equality of the objects at the end of paths Pf_1 and Pf_2, and predicates $\mathsf{C}_{C'}(\mathsf{Pf})$ express that the object at the end of path Pf is in the interpretation of concept C'.

Our PTIME result for the *partial*$-\mathcal{CFD}$ implication problem follows by a simple check for goals occurring in a ground model for expansion rules generated by a polynomial sized collection of path expressions.

Theorem 7. Let $\mathcal{T}$ be a *partial*$-\mathcal{CFD}$ terminology and $\mathcal{Q}$ a posed question of the form $E_1 \sqsubseteq E_2$. Then

$$\mathcal{T} \models \mathcal{Q} \text{ iff } \mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id), \mathsf{D}(id)\} \models \mathsf{G}_{E_1} \text{ or}$$
$$\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id), \mathsf{D}(id)\} \models \mathsf{G}_{\forall\, \mathsf{Pf}\,.\,\bot}(id) \text{ for some } \mathsf{Pf} \in \mathsf{PF}(\mathcal{T}, \mathcal{Q}).$$

Proof (sketch): If $\mathsf{C}_\bot(\mathsf{Pf})$ and $\mathsf{D}(\mathsf{Pf})$ for $\mathsf{Pf} \in \mathsf{PF}(\mathcal{T}, \mathcal{Q})$ appear in M, where M is the least model of $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\}$ $\mathsf{R}(\mathcal{T})$, then the concept E_1 is unsatisfiable w.r.t. $\mathcal{T}$ since only implied facts appear in M, and therefore the subsumption holds for any E_1 and $\mathcal{T}$.

Otherwise, if $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\} \not\models \mathsf{G}_{E_1}$, then there must be a model M of $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\}$ such that $G \notin M$ for some $G \in \mathsf{G}_{E_1}$. We construct an interpretation $\mathcal{I}_M$ such that $\mathcal{I}_M \models \mathcal{T}$ but $\mathcal{I}_M \not\models \mathcal{Q}$. The interpretation $\mathcal{I}_M$ contains an object o for each equivalence class defined on the set $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ by the interpretation of E. The class membership of these objects is determined by the membership of the corresponding path in the interpretations of the C_C predicates in M. Note that, due to the syntactic restriction imposed on PFDs, this is sufficient to satisfy all PFDs in $\mathcal{T}$ since any precondition or a non-trivial consequence of a PFD can only manifest on some path belonging to $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ and beginning at the distinguished object o. To complete the construction of $\mathcal{I}_M$, we simply attach a unique complete tree F^* to each leaf node (i.e., a node that is missing successors). Nodes of these complete trees belong to all primitive descriptions in $\mathcal{I}_M$ and thus satisfy $\mathcal{T}$.

Conversely, assume $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\} \models \mathsf{G}_{E_1}$ but $\mathcal{T} \not\models \mathcal{Q}$. Then there must be an interpretation $\mathcal{I}$ and an object $o \in \triangle$ such that $\mathcal{I} \models \mathcal{T}$ and $o \in E_1^{\mathcal{I}} - E_2^{\mathcal{I}}$. Thus, there is a model $M_{\mathcal{I}}$ of $\mathsf{R}(\mathcal{T})$ such that $\mathsf{C}_{E_1}(id) \in M_{\mathcal{I}}$. In this model, the element $id \in \mathsf{PF}(\mathcal{T}, \mathcal{Q})$ serves as the counterpart of the object o and the interpretations of the predicates C_C and E is *extracted* from $\mathcal{I}$ by navigating all (pairs of) path functions in $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$. However, since $o \notin E_2^{\mathcal{I}}$, it must be the case that $M_{\mathcal{I}}$ is a strict subset of the least model of $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\}$; a contradiction. □

Since the expansion rules are Horn clauses over a finite universe $\mathsf{PF}(\mathcal{T}, \mathcal{Q})$ of polynomial size, we have the following:

Corollary 8. Let $\mathcal{T}$ be a terminology and $\mathcal{Q}$ a posed question in *partial*$-\mathcal{CFD}$. Then the implication problem $\mathcal{T} \models \mathcal{Q}$ is complete for PTIME.

Proof (sketch): The least model of $\mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id)\}$ can be obtained by using a bottom-up construction of the least fix-point of the rules in time polynomial in $|\mathcal{T}| + |\mathcal{Q}|$ (since all predicates in $\mathsf{R}(\mathcal{T})$ have a fixed arity). Hardness follows from embedding Horn-SAT into reasoning with PFDs. □

In practice, elements of this set can be constructed on demand by using additional Horn rules in such a way that only path expressions needed to confirm subsumption or non-subsumption are generated [4].

Note that neither *partial*$-\mathcal{CFD}$ nor $\mathcal{CFD}$ can be extended to allow disjointness (bottom ($\bot$)), negation (hence $\neg\exists f$ cannot be used), or disjunction on the right hand sides of inclusion dependencies while maintaining PTIME decidability of logical implication [8]. However, a similar technique as in the above development can be used to handle partial features without impacting the complexity of the logical implication problems.

4.2 *partial*$-\mathcal{CFD}^{\forall}_{nc}$

partial$-\mathcal{CFD}^{\forall}_{nc}$ shares the PFD restrictions with *partial*$-\mathcal{CFD}$. However, it trades the ability to use conjunctions on the left hand sides of TBox inclusion dependencies for the ability to express disjointness and conditional typing:

1. left hand sides must be primitive concepts or value restrictions, and
2. right hand sides must be a primitive concepts, negations of primitive concepts, conjunctions, value restrictions, existential restrictions, and PFDs (again, restricted as in (2)).

It is easy to see that every *partial*$-\mathcal{CFD}^{\forall}_{nc}$ TBox $\mathcal{T}$ is consistent (by setting all primitive concepts to be interpreted as the empty set). It is, however, no longer true that all primitive concepts (and their conjunctions) are trivially satisfiable. For example, $\mathrm{A} \sqsubseteq \neg\mathrm{A} \in \mathcal{T}$ forces A to be empty in every model of $\mathcal{T}$.

Concept Satisfiability. The problem of *concept satisfiability* asks, for a given concept C and TBox $\mathcal{T}$, if there exists an interpretation $\mathcal{I}$ for $\mathcal{T}$ in which $C^{\mathcal{I}}$ is non-empty. Such problems can be reduced to the case where C is a primitive concept A by simply augmenting $\mathcal{T}$ with $\{\mathrm{A} \sqsubseteq C\}$, where A is a fresh primitive concept. Note that concept C can be a *conjunction* of other concepts since it only appears on the right-hand side of an inclusion dependency. We proceed as follows:

Definition 9 (Transition Relation for $\mathcal{T}$). Let $\mathcal{T}$ be a *partial*$-\mathcal{CFD}^{\forall}_{nc}$ TBox in normal form. We define a transition relation $\delta(\mathcal{T})$ over the set of states

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

and the alphabet F as follows:

$$C_1 \xrightarrow{id} C_2 \in \delta(\mathcal{T}), \text{ if } C_1 \sqsubseteq C_2 \in \mathcal{T}, \text{ and}$$
$$\forall f.\mathrm{A} \xrightarrow{f} \mathrm{A} \in \delta(\mathcal{T}), \text{ if } \forall f.\mathrm{A} \xrightarrow{id\,*} \exists f \in \delta(\mathcal{T}),$$

where *id* is the empty letter transition, $id*$ is a sequence of *id* edges, $f \in \mathsf{F}$, $A \in \mathsf{PC}$, and $C_1, C_2 \in S$. □

The transition relation allows us to construct *non-deterministic finite automata* (NFA) that can be used for various reasoning problems on a *partial*$-\mathcal{CFD}^{\forall}_{nc}$ TBox $\mathcal{T}$. Note that, unlike common practice in automata theory, we use *id* for the empty letter in transition relations. Given a primitive concept A and TBox $\mathcal{T}$, one can test for primitive concept satisfiability by using the following NFA, denoted $\mathsf{nfa}^{\mathrm{A}}_{\mathrm{B}}(\mathcal{T})$:

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

with states induced by primitive concepts, their negations, and value restrictions, with start state A, with the set of final states $\{\mathrm{B}\} \subseteq S$, and with transition relation $\delta(\mathcal{T})$. Intuitively, if $\mathsf{Pf} \in \mathsf{nfa}^{\mathrm{A}}_{\mathrm{B}}(\mathcal{T})$ and $o \in \mathrm{A}^{\mathcal{I}}$ then $\mathsf{Pf}^{\mathcal{I}}(o)$ is defined and $\mathsf{Pf}^{\mathcal{I}}(o) \in \mathrm{B}^{\mathcal{I}}$ in every model $\mathcal{I}$ of $\mathcal{T}$.

Theorem 10 (Concept Satisfiability). A is satisfiable with respect to the TBox $\mathcal{T}$ if and only if

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

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

Proof (sketch): Assume A is non-empty and hence there is $a \in \mathrm{A}^{\mathcal{I}}$. For a primitive concept $\mathrm{B} \in \mathsf{PC}$, a word Pf in the intersection language of the two automata above is a witness of the fact that $\mathsf{Pf}^{\mathcal{I}}(a^{\mathcal{I}}) \in \mathrm{B}^{\mathcal{I}}$ and $\mathsf{Pf}^{\mathcal{I}}(a^{\mathcal{I}}) \in \neg\mathrm{B}^{\mathcal{I}}$ must hold in every model of $\mathcal{T}$.

Conversely, if no such word exists, then one can construct a *deterministic* finite automaton from $\mathsf{nfa}^{\mathrm{A}}_{\mathrm{B}}(\mathcal{T})$, using the standard subset construction, in which there is not a state containing both B and ¬B reachable from the start state A. Unfolding the transition relation of this automaton, starting from the state A and labelling nodes by the concepts associated with the automaton's states, yields a tree interpretation that satisfies $\mathcal{T}$ (in particular in which all PFD constraints are satisfied vacuously) and whose root provides a witness for satisfiability of A. □

To test for emptiness of $\mathsf{nfa}^{\mathrm{A}}_{\mathrm{B}}(\mathcal{T})$, we use a graph connectivity algorithm that non-deterministically searches for a $(\mathrm{A}, \mathrm{A}) - (\mathrm{B}, \neg\mathrm{B})$ path in the (virtual) poly-sized product automaton [2]; the following result is then immediate.

Corollary 11. Concept satisfiability with respect to *partial*$-\mathcal{CFD}^{\forall}_{nc}$ TBoxes is complete for NLOGSPACE. □

Note that, as we remarked above, this procedure can be used to test for satisfiability of *conjunctions of concepts* in $\mathcal{CFD}^{\forall}_{nc}$ as follows:

Lemma 12. $\mathrm{A}_1 \sqcap \ldots \sqcap \mathrm{A}_k$ is consistent in $\mathcal{T}$ if and only if A is satisfiable in $\mathcal{T} \cup \{\mathrm{A} \sqsubseteq \mathrm{A}_i \mid 1 \leq i \leq k\}$. □

It is, however, impossible to *precompute* all such inconsistent concepts since this would require consideration of all possible *types* over PC (or finite subsets of primitive concepts), a process essentially equivalent to constructing an equivalent deterministic automaton which can require exponential time [2].

Logical Implication. Logical implication for *partial*$-\mathcal{CFD}^{\forall}_{nc}$ TBoxes $\mathcal{T}$ and posed questions $\mathcal{Q}$ can now be solved similarly to the $\mathcal{CFD}$ case. The main difference lies in detecting inconsistencies caused by object membership in conjunctions of (primitive) concepts that are necessarily *empty* in models of $\mathcal{T}$. This observation yields the following extension to $\mathsf{R}(\mathcal{T})$:

> If $\mathsf{D}(\mathsf{Pf})$ and $\mathsf{C}_{\mathrm{A}_1}(\mathsf{Pf}), \ldots, \mathsf{C}_{\mathrm{A}_k}(\mathsf{Pf})$ are in $\mathsf{R}(T)$ for some $\mathsf{Pf} \in \mathsf{PF}(\mathcal{T}, \mathcal{Q})$ and $\mathrm{A}_1 \sqcap \ldots \sqcap \mathrm{A}_k$ is not consistent in $\mathcal{T}$ then add $\mathsf{C}_{\perp}(\mathsf{Pf})$ to $\mathsf{R}(\mathcal{T})$.

Note that *partial*$-\mathcal{CFD}^{\forall}_{nc}$ can be extended to allow $\neg\exists f$ on the right hand sides of TBox dependencies (which would be handled analogously to negated primitive concepts by the NFA in Theorem 10). Alltogether, we obtain following results analogous to those in Sect. 4.1:

Theorem 13. Let $\mathcal{T}$ be a *partial*$-\mathcal{CFD}^{\forall}_{nc}$ terminology and $\mathcal{Q}$ a posed question of the form $E_1 \sqsubseteq E_2$. Then

$$\begin{aligned} \mathcal{T} \models \mathcal{Q} \text{ iff } & \mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id), \mathsf{D}(id)\} \models \mathsf{G}_{E_1} \text{ or} \\ & \mathsf{R}(\mathcal{T}) \cup \{\mathsf{C}_{E_1}(id), \mathsf{D}(id)\} \models \mathsf{G}_{\forall \mathsf{Pf}.\perp}(id) \text{ for some } \mathsf{Pf} \in \mathsf{PF}(\mathcal{T}, \mathcal{Q}). \end{aligned}$$

Proof (sketch): The proof is similar to the proof of Theorem 7: the necessary F^* trees are generated by unfolding $\delta(\mathcal{T})$ as in the Proof of Theorem 10. □

Corollary 14. Let $\mathcal{T}$ be a terminology and $\mathcal{Q}$ a posed question in *partial*$-\mathcal{CFD}^{\forall}_{nc}$. Then the implication problem $\mathcal{T} \models \mathcal{Q}$ is complete for PTIME. □

5 Summary and Future Work

In summary, we have shown how partial features coupled with a strict interpretation of undefined values can be incorporated in the feature-based DLs $\mathcal{DLFD}$, $\mathcal{CFD}$ and $\mathcal{CFD}^{\forall}_{nc}$, thus obtaining *partial*$-\mathcal{DLFD}$, *partial*$-\mathcal{CFD}$ and *partial*$-\mathcal{CFD}^{\forall}_{nc}$, respectively. Our primary contributions have been to also show that this can be done without impact on the complexity of their associated logical inference problems. Indeed, with $\mathcal{DLFD}$, this was achieved by showing how *partial*$-\mathcal{DLFD}$ can be fully simulated in $\mathcal{DLFD}$ in an entirely transparent fashion.

One avenue for future work would be to consider the impact on logical inference of alternative semantics for the interpretation of undefined values, in particular, on choosing the so-called Kleene semantics for equality. In this case, "$e_1 = e_2$" is also true when both e_1 and e_2 have undefined values. Note that doing so with $\mathcal{DLFD}$ would in fact *necessitate* changes to the semantics of the PFD concept constructor to avoid undecidability of logical inference.[4]

In our introductory comments, we also hinted at possible directions for future work relating to $\mathcal{CFD}^{\forall}_{nc}$. The first concerns recent work that begins to explore

[4] The details for this are beyond the scope of the paper.

how *inverse features* can be added to feature-based DLs, in particular, on adding the $\exists f^{-1}$ concept constructor. For example, logical inference has been shown to be decidable in PTIME for $\mathcal{CFDI}_{nc}^{\forall -}$ [11], a dialect obtained by adding this constructor to $\mathcal{CFD}_{nc}^{\forall}$ and by imposing additional syntactic restrictions, e.g., on the syntax of PFDs, to avoid intractability for this problem. We conjecture that our results for $partial{-}\mathcal{CFD}_{nc}^{\forall}$ can be extended to $partial{-}\mathcal{CFDI}_{nc}^{\forall -}$, although the development would be much less straightforward, and that the same applies to the other two dialects that we have considered: that logical consequence for $partial-\mathcal{CFDI}$ and $partial-\mathcal{DLFDI}$ is decidable in PTIME and EXPTIME, respectively.

The second possible direction for future work relating to $\mathcal{CFD}_{nc}^{\forall}$ is an indirect consequence of the ability to easily simulate $partial{-}\mathcal{DLFD}$ in $\mathcal{DLFD}$. In particular, for this case, there remains little incentive to adopt $partial{-}\mathcal{DLFD}$: to non-trivially complicate the semantics of feature-based DLs, to add the existential restriction concept constructor, and so on. We conjecture that it is possible to extend the syntax of $\mathcal{CFD}_{nc}^{\forall}$ to allow limited use of conjunction on left-hand-sides of inclusion dependencies to enable simulating $partial{-}\mathcal{CFD}_{nc}^{\forall}$ in $\mathcal{CFD}_{nc}^{\forall}$ in an analogous fashion, while preserving PTIME decidability for logical consequence.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Boston (1979)
3. Khizder, V.L., Toman, D., Weddell, G.: Reasoning about duplicate elimination with description logic. In: Rules and Objects in Databases (DOOD, part of CL 2000), pp. 1017–1032 (2000)
4. Ramakrishnan, R.: Magic templates: a spellbinding approach to logic programs. J. Logic Program. **11**(3 & 4), 189–216 (1991)
5. Toman, D., Weddell, G.: On attributes, roles, and dependencies in description logics and the ackermann case of the decision problem. In: Description Logics 2001, CEUR-WS, vol. 49, pp. 76–85 (2001)
6. Toman, D., Weddell, G.: On reasoning about structural equality in XML: a description logic approach. Theor. Comput. Sci. **336**(1), 181–203 (2005)
7. Toman, D., Weddell, G.E.: On keys and functional dependencies as first-class citizens in description logics. J. Aut. Reason. **40**(2–3), 117–132 (2008)
8. Toman, D., Weddell, G.E.: Applications and extensions of PTIME description logics with functional constraints. In: Proceedings International Joint Conference on Artificial Intelligence (IJCAI), pp. 948–954 (2009)
9. Toman, D., Weddell, G.E.: Conjunctive query answering in $\mathcal{CFD}_{nc}$: a PTIME description logic with functional constraints and disjointness. In: AI 2013: Advances in Artificial Intelligence - 26th Australasian Joint Conference, Dunedin, New Zealand, pp. 350–361 (2013)

10. Toman, D., Weddell, G.E.: Answering queries over $\mathcal{CFD}_{nc}^{\forall}$ knowledge bases. Technical report CS-2014-14, Cheriton School of Computer Science, University of Waterloo (2014)
11. 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, vol. 8862, pp. 587–599. Springer, Heidelberg (2014)