On Decidability and Complexity of Description Logics with Uniqueness Constraints*

Vitaliy L. Khizder, David Toman, and Grant Weddell

Department of Computer Science
University of Waterloo, Canada

Abstract. We establish the equivalence of: (1) the logical implication problem for a description logic dialect called DLClass that includes a concept constructor for expressing uniqueness constraints, (2) the logical implication problem for path functional dependencies (PFDs), and (3) the problem of answering queries in deductive databases with limited use of successor functions. As a consequence, we settle an open problem concerning lower bounds for the PFD logical implication problem and show that a regularity condition for DLClass that ensures low order polynomial time decidability for its logical implication problem is tight.

1 Introduction

Description Logics (DLs) have many applications in information systems [2]. They can facilitate data access to heterogenous data sources because of their ability to capture integrity constraints manifest in object relational database schema, in ER diagrams or UML class diagrams and that arise in practical XML applications, including the constraints that underly XML *document type definitions* (DTDs) [10]. They are particularly valuable for solving problems in information integration and that arise in query optimization [4, 8, 16, 17]. However, in many of these applications, it becomes essential for a particular DL dialect to capture knowledge that relates to various kinds of *uniqueness constraints* that are satisfied by the possible data sources.

For example, consider a hypothetical integrated patient management system depicted in Figure 1. It can be crucial to know for this system

1. that *a hospital has a unique name*,
2. that *a patient is uniquely identified by hospital and patient number*, and
3. that *a person admitted to a hospital has a valid unique social security number*.

DL dialects that enable capturing keys in the standard database sense and simple forms of functional dependencies have been proposed in [4, 7]. Such a facility was a nagging missing ingredient for prior DL dialects that had efficient subsumption

* An earlier version of this paper has appeared in the informal proceedings of the International Workshop on Description Logics DL2000.

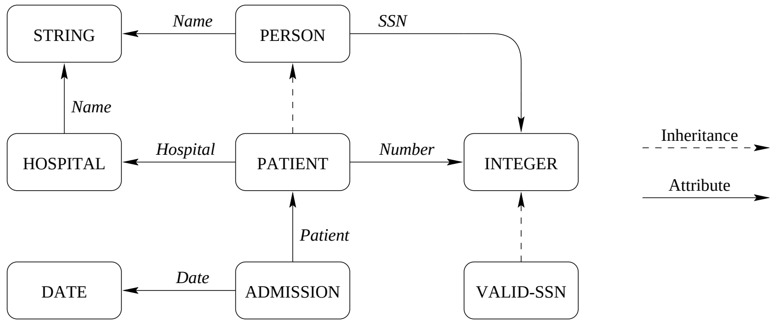

Fig. 1. Patient Data Integration.

checking algorithms. This was achieved by adding a new kind of *concept constructor* (fd). Also, Calvanese et al. have recently demonstrated how a variety of keys with set-valued components can be added to a very general DL dialect without increasing the complexity of reasoning [9].

In this paper, we consider a more general version of fd in which component attribute descriptions may now correspond to *attribute* or *feature paths* [15–17, 20]. To focus on the essential idea, we define a very simple DL dialect called DLFD that consists of a single concept constructor corresponding to this more general version of fd. The logical implication problem for DLFD is therefore a special case of the logical implication problem for *path functional dependencies* (PFDs), a variety of uniqueness constraints for data models supporting complex objects that was first proposed in [20] and studied more fully in a subsequent series of papers [15, 19, 21].

Although DLFD is extremely simple, it can be used to simulate logical implication problems in a more general *object relational* dialect via a linear translation. DLClass, as we shall call it, includes additional concept constructors that directly capture class inheritance and attribute typing, a capability that is essential to the convenient capture of meta-data relating to possible data sources in information systems.

For example, consider again the hospital schema in Figure 1 for the abovementioned patient management system. The schema captures many of the constraints that underly an information source that contains XML data with the structure illustrated in Figure 2. The schema, perhaps given more directly by an associated DTD for the data, can be captured in DLClass as a *terminology* that consists of the following *subsumption constraints* (cf. Definition 1).

$$
\begin{aligned}
\text{ADMISSION} &< (\mathsf{and}\ (\mathsf{all}\ \mathit{Date}\ \text{DATE})\ (\mathsf{all}\ \mathit{Patient}\ \text{PATIENT}))\\
\text{PATIENT} &< (\mathsf{all}\ \mathit{Hospital}\ \text{HOSPITAL})\\
\text{PATIENT} &< (\mathsf{and}\ \text{PERSON}\ (\mathsf{all}\ \mathit{Number}\ \text{INTEGER}))\\
\text{PERSON} &< (\mathsf{and}\ (\mathsf{all}\ \mathit{Name}\ \text{STRING})\ (\mathsf{all}\ \mathit{SSN}\ \text{INTEGER}))\\
\text{HOSPITAL} &< (\mathsf{all}\ \mathit{Name}\ \text{STRING})\\
\text{VALID-SSN} &< \text{INTEGER}
\end{aligned}
$$

```
<ADMISSION>
   <PATIENT Number=123>
      <PERSON SSN=45678>
         <NAME>Fred</NAME>
      </PERSON>
      <HOSPITAL>
         <NAME>Sunny Brook</NAME>
      </HOSPITAL>
   </PATIENT>
   <DATE>20-January-2000</DATE>
</ADMISSION>
```

Fig. 2. Patient Data in XML.

Furthermore, DLClass can be used to capture the three uniqueness constraints for the patient management system listed at the beginning of this section by adding the following additional subsumption constraints to the terminology.

$$\begin{aligned}
\text{HOSPITAL} &< (\mathsf{fd}\ \ \text{HOSPITAL} : \mathit{Name} \to \mathit{Id}) \\
\text{PATIENT} &< (\mathsf{fd}\ \ \text{PATIENT} : \mathit{Hospital}, \mathit{Number} \to \mathit{Id}) \\
\text{ADMISSION} &< (\mathsf{all}\ \mathit{Patient}\ (\mathsf{and}\ (\mathsf{all}\ \mathit{SSN}\ \ \text{VALID-SSN}) \\
&\qquad\qquad\qquad\qquad (\mathsf{fd}\ \ \text{PERSON} : \mathit{SSN} \to \mathit{Id})))
\end{aligned}$$

Again, we show that DLFD is sufficiently powerful to simulate all inferences in DLClass by exhibiting a linear answer preserving translation of DLClass to DLFD. However, this translation is indirect: it relates the logical implication problems in DLFD and DLClass to query answering in Datalog$_{nS}$, a deductive query language with limited use of successor functions [11, 12]. This relationship leads to the main contribution of the paper; an open issue relating to the complexity of reasoning about PFDs is resolved. By proving an equivalence to query answering in Datalog$_{nS}$, the logical implication problems for DLFD and DLClass are DEXPTIME-complete, and therefore the exponential time decision procedure [15] becomes tight.

The example constraints in DLClass given above have a restricted form that satisfies a syntactic *regularity* condition (cf. Section 5), which leads to incremental polynomial time algorithms for a restricted class of logical implication problems in DLClass [16]. Using the tight translation between DLClass and Datalog$_{nS}$ we show that a similar condition can be applied to Datalog$_{nS}$ programs. This leads to a PTIME query evaluation procedure for a syntactically restricted class of Datalog$_{nS}$ programs. In addition, the condition turns out to be as general as one can hope for while ensuring the existence of such efficient algorithms.

The remainder of the paper is organized as follows. Section 2 defines the syntax and semantics for DLClass and Datalog$_{nS}$. Section 3 reduces the problem of answering queries in Datalog$_{nS}$ to the logical implication problem for DLFD and subsequently to logical implication problems in DLClass. Section 4 completes the picture by reducing the logical implication problem for DLClass to Datalog$_{nS}$. Section 5 discusses special cases in which PTIME reasoning is possible. We conclude with a summary in Section 6.

2 Definitions

The syntax and semantics of DLClass and Datalog$_{nS}$ are given by the following.

Definition 1 (Description Logic DLClass) *Let F be a set of attribute names. We define a* path expression *by the grammar "*$\mathsf{Pf} ::= f.\,\mathsf{Pf} \mid Id$*" for $f \in F$. Let C be primitive concept description(s). We define derived* concept descriptions *using the following grammar:*

$$
\begin{aligned}
D ::= {} & C \\
\mid {} & (\mathsf{all}\ f\ D) \\
\mid {} & (\mathsf{fd}\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}), k > 0 \\
\mid {} & (\mathsf{and}\ D\ D)
\end{aligned}
$$

A subsumption constraint *is an expression of the form $C < D$.*

The semantics *of expressions is given with respect to a structure $(\Delta, .^I)$, where Δ is a domain of "objects" and $.^I$ an interpretation function, that fixes the interpretations of primitive concepts to be subsets of the domain, $C^I \subseteq \Delta$, and primitive attributes to be total functions on the domain, $f^I : \Delta \rightarrow \Delta$. This interpretation is extended to path expressions, $Id^I = \lambda x.x$ and $f.\,\mathsf{Pf}^I = \mathsf{Pf}^I \circ f^I$, and to derived descriptions*

$$
\begin{aligned}
(\mathsf{all}\ f\ D)^I &= \{o \in \Delta : f^I(o) \in D^I\} \\
(\mathsf{fd}\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf})^I &= \{o \in \Delta : \forall o' \in C^I. \\
&\qquad \textstyle\bigwedge_{i=1}^{k} \mathsf{Pf}_i^I(o) = \mathsf{Pf}_i^I(o') \Rightarrow \mathsf{Pf}^I(o) = \mathsf{Pf}^I(o')\} \\
(\mathsf{and}\ D_1\ D_2)^I &= D_1^I \cap D_2^I
\end{aligned}
$$

An interpretation satisfies a subsumption constraint *$C < D$ if $C^I \subseteq D^I$.*

For a given set of subsumption constraints $\Sigma = \{C_i < D_i : 0 < i \leq n\}$ (a terminology*) and a subsumption constraint $C < D$ (a* posed question*), the* logical implication problem *asks if $\Sigma \models C < D$ holds, i.e, if all interpretations that satisfy Σ must also satisfy $C < D$.*

Limiting the left-hand-side of subsumption constraints in terminologies to be primitive concepts is a common assumption to avoid reasoning about equality between general concept descriptions. In contrast, requiring the left-hand-side of the posed question to be a primitive concept is not a real limitation since a more general logical implication problem of the form $\Sigma \models D_1 < D_2$ can always be rephrased as $\Sigma \cup \{C < D_1\} \models C < D_2$, where C is a primitive concept not occurring in $\Sigma \cup \{D_1 < D_2\}$.

In the rest of the paper we simplify the notation for path expressions by omitting the trailing Id. We also allow a syntactic composition $\mathsf{Pf}_1 . \mathsf{Pf}_2$ of path expressions that stands for their concatenation.

Definition 2 (Datalog$_{nS}$ [12]) *Let p_i be predicate symbols, f_i function symbols such that $p_i \neq f_i$, and $X, Y, \dots$ variables. A* logic program *P is a finite set of Horn clauses of the form*

$$p_0(t^0, s_1^0, \dots, s_k^0) \leftarrow p_1(t^1, s_1^1 \dots, s_{l_1}^1), \dots, p_k(t^k, s_1^k, \dots, s_{l_k}^k)$$

for $k \geq 0$, where the terms t^i and s_j^i are constructed from constants, function symbols and variables. We say that P is a Datalog$_{nS}$ program *if*

1. *t^i is a* functional term*: a variable, a distinguished constant 0, or a term of the form $f(t, s_1, \dots s_l)$ where f is a function symbol, t is a functional term, and $s_1, \dots, s_l$ are data terms,*
2. *s_j^i are* data terms*: variables or constants different from 0, and*
3. *no variable appears both in a functional and a data term.*

We say that a Datalog$_{nS}$ program is in normal form *if the only predicate and functions symbols used are unary, and whenever a variable appears in any predicate p_i of a clause then the same variable appears in all the predicates of the same clause.*

A recognition problem *for a Datalog$_{nS}$ program P and a ground (variable-free) atom $q(t, s_1, \dots, s_k)$ is the question does $P \models q(t, s_1, \dots, s_k)$ hold, i.e., $q(t, s_1, \dots, s_k)$ is true in all models of P?*

It is known that every Datalog$_{nS}$ program can be encoded as a normal Datalog$_{nS}$ program [11, 12]. Moreover, every normal Datalog$_{nS}$ program can be divided into a set of clauses with non-empty bodies and a set of ground *facts* (clauses with empty bodies). Proofs of theorems in the paper rely on the following two observations about logic programs [18]:

1. The recognition problem $P \models q$ for a ground q is equivalent to checking $q \in M_P$ where M_P is the unique *least Herbrand model* of P.
2. M_P can be constructed by iterating an *immediate consequence operator* T_P associated with P until reaching fixpoint; $M_P = T_P^\omega(\emptyset)$.

To establish the complexity bounds we use the following result about Datalog$_{nS}$ programs:

Proposition 3 ([12, 14]) *The recognition problem for Datalog$_{nS}$ programs (under the data-complexity*[1] *measure) is DEXPTIME-complete. The lower bound holds even for programs in normal form.*

[1] Complexity of the problem for a fixed set of symbols.

3 Lower Bounds

In this section we show that the recognition problem for Datalog$_{nS}$ can be reduced to the (infinite) implication problem for path-functional dependencies. We study this problem in a DL dialect DLFD in which all subsumption constraints are of the form

$$\mathsf{THING} < (\mathsf{fd}\ \mathsf{THING} : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}).$$

THING is a primitive concept interpreted as the domain Δ. In the rest of this section we use the shorthand $\mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}$ for the above constraint.

It is easy to see that DLFD problems can be trivially embedded into DLClass. We simply consider THING to be a single primitive concept description such that $\mathsf{THING} < (\mathsf{all}\ f\ \mathsf{THING})$ for every primitive attribute f. Therefore, lower bounds for DLFD also apply to DLClass.

Notation: we associate two Datalog$_{nS}$ functional terms $\overline{\mathsf{Pf}}(0) = f_k(\cdots f_1(0)\cdots)$ and $\overline{\mathsf{Pf}}(X) = f_k(\cdots f_1(X)\cdots)$ with every path expression $\mathsf{Pf} = f_1.\cdots.f_k.Id$, where 0 is a distinguished constant and X is a variable. Similarly, for every Datalog$_{nS}$ term $t = f_1(\cdots f_k(X)\cdots)$ there is a path expression $\mathsf{Pf} = f_k.\cdots.f_1.Id$ such that $t = \overline{\mathsf{Pf}}(X)$. In the rest of the paper we overload the symbols p_i and f_i to stand both for unary predicate and function symbols in Datalog$_{nS}$ and for primitive attribute names in the appropriate description logic, and use $\overline{\mathsf{Pf}}(X)$ and $\overline{\mathsf{Pf}}(0)$ to stand for Datalog$_{nS}$ terms.

Theorem 4 *Let P be an arbitrary normal Datalog$_{nS}$ program and $G = p(\overline{\mathsf{Pf}}(0))$ a ground atom. We define*

- $\Sigma_P = \{\mathsf{Pf}'_1 .p'_1, \ldots, \mathsf{Pf}'_k .p'_k \rightarrow \mathsf{Pf}' .p' :$
  $p'(\overline{\mathsf{Pf}}'(X)) \leftarrow p'_1(\overline{\mathsf{Pf}}'_1(X)), \ldots, p'_k(\overline{\mathsf{Pf}}'_k(X)) \in P\}$, *and*
- $\varphi_{P,G} = \mathsf{Pf}_1 .p_1, \ldots, \mathsf{Pf}_k .p_k \rightarrow \mathsf{Pf} .p$
  where $p_1(\overline{\mathsf{Pf}}_1(0)), \ldots, p_k(\overline{\mathsf{Pf}}_k(0))$ *are all facts in* P.

Then $P \models G \iff \Sigma_P \models \varphi_{P,G}$.

<u>Proof:</u> $\Rightarrow$: We show that $p(\overline{\mathsf{Pf}}(0)) \in T_P^{\omega}(\emptyset)$ implies $\Sigma_P \models \varphi_{P,G}$.

If $p(\overline{\mathsf{Pf}}(0)) \in T_P^{\omega}(\emptyset)$ then there must be $m > 0$ such that $p(\overline{\mathsf{Pf}}(0)) \in T_P^{m}(\emptyset)$. Then, by induction on m, we have:

$m = 1$: immediate as it must be the case that $p(\overline{\mathsf{Pf}}(0))$ must be one of the facts in P, e.g., $p_i(\overline{\mathsf{Pf}}_i(0))$ and therefore also $\mathsf{Pf} .p = \mathsf{Pf}_i .p_i$. Consequently, $\Sigma_P \models \varphi_{P,G}$ as $\varphi_{P,G}$ is a trivial path-functional dependency.

$m > 1$: if $p(\overline{\mathsf{Pf}}(0)) \in T_P^{m}(\emptyset)$ then there is a term $t(0)$ (to be substituted for X) and A clause $p(\overline{\mathsf{Pf}}'(X)) \leftarrow p_1(\overline{\mathsf{Pf}}'_1(X)), \ldots, p_l(\overline{\mathsf{Pf}}'_l(X))$ in P such that $\overline{\mathsf{Pf}}(0) = \overline{\mathsf{Pf}}'(t(0))$ and $p_i(\overline{\mathsf{Pf}}'_i(t(0))) \in T_P^{m-1}(\emptyset)$. By the IH we have $\Sigma_P \models \varphi_{P,p_i(\overline{\mathsf{Pf}}'_i(t(0)))}$ and therefore, by composition with the path-functional dependency $\mathsf{Pf}'_1 .p_1, \ldots, \mathsf{Pf}'_l .p_l \rightarrow \mathsf{Pf} .p \in \Sigma_P$ we have $\Sigma_P \models \varphi_{P,G}$.

$\Leftarrow$: Assume $p(\overline{\mathsf{Pf}}(0)) \notin T_P^\omega(\emptyset)$. We construct an counterexample interpretation as follows: let $o_1, o_2 \in \Delta$ be two distinct objects and T_1, T_2 two complete infinite trees rooted by these two objects with edges labeled by primitive attributes, in this case f_i and p_i. Moreover, if $p'(\overline{\mathsf{Pf}}'(0)) \in T_P^\omega(\emptyset)$ we merge the two subtrees identified by the path $\mathsf{Pf}'.p'$ starting from the respective roots of the trees. The resulting graph provides an interpretation for DLFD (the nodes of the trees represent elements of Δ, the edges give interpretation to primitive attributes) such that:

(i) $(\mathsf{Pf}_i.p_i)^I(o_1) = (\mathsf{Pf}_i.p_i)^I(o_2)$ for all $p_i(\overline{\mathsf{Pf}}_i(0)) \in P$, and

(ii) every constraint in Σ_P is satisfied by the constructed interpretation: Assume the interpretation violated a constraint $\mathsf{Pf}'_1.p'_1, \ldots, \mathsf{Pf}'_l.p'_l \to \mathsf{Pf}'.p' \in \Sigma_P$. Then there must be two distinct elements x_1, x_2 such that $(\mathsf{Pf}'_i.p'_i)^I(x_1) = (\mathsf{Pf}'_i.p'_i)^I(x_2)$. From the construction of the interpretation and the fact that the sets of predicate and function symbols are disjoint we know that $p'_i(\overline{\mathsf{Pf}}'_i(t(0))) \in T_P^\omega(\emptyset)$ where t is a term corresponding to the paths from o_1 and o_2 to x_1 and x_2, respectively (note that all the paths that end in a particular common node in the constructed interpretation are symmetric). However, then $p'(\overline{\mathsf{Pf}}'(t(0))) \in T_P^\omega(\emptyset)$ using the clause in P associated with the violated constraint in Σ_P, and thus $(\mathsf{Pf}'.p')^I(x_1) = (\mathsf{Pf}'.p')^I(x_2)$, a contradiction.

On the other hand, $(\mathsf{Pf}.p)^I(o_1) \neq (\mathsf{Pf}.p)^I(o_2)$ as $p(\overline{\mathsf{Pf}}(0)) \notin T_P^\omega(\emptyset)$, a contradiction. $\square$

For the constructed DLFD problem we have $|\Sigma_P| + |\varphi_{P,G}| \in \mathcal{O}(|P| + |G|)$. Thus:

Corollary 5 *The logical implication problem for DLFD is DEXPTIME-hard.*

Since DLFD problems can be embedded into DLClass, we have:

Corollary 6 *The logical implication problem for DLClass is DEXPTIME-hard.*

4 Upper Bound and Decision Procedure for DLClass

To complete the picture we exhibit a DEXPTIME decision procedure for DLClass by reducing an arbitrary logical implication problem to the recognition problem for Datalog$_{nS}$ [12]. We start with two lemmas that are used to simplify complex DLClass constraints.

Lemma 7 *Let C_1 be a primitive concept not in $\Sigma \cup \{C' < D'\}$. Then*

$$\Sigma \cup \{C < (\mathsf{and}\ D_1\ D_2)\} \models C' < D' \iff \Sigma \cup \{C < D_1, C < D_2\} \models C' < D',$$

$$\Sigma \cup \{C < (\mathsf{all}\ f\ D)\} \models C' < D' \iff \Sigma \cup \{C < (\mathsf{all}\ f\ C_1), C_1 < D\} \models C' < D'.$$

We say that a terminology Σ is *simple* if it does not contain descriptions of the form (and D_1 D_2) and whenever (all f D) appears in Σ then D is a primitive concept description. Lemma 7 shows that every terminology can be converted to an equivalent simple terminology.

Lemma 8 *Let Σ be a simple terminology and C_1 a primitive concept not present in $\Sigma \cup \{C < D, C' < D'\}$. Then*

$$\Sigma \models C < (\mathsf{and}\ D_1\ D_2) \iff \Sigma \models C < D_1 \text{ and } \Sigma \models C < D_2,$$

$$\begin{aligned}\Sigma \models C < (\mathsf{all}\ f\ D) \iff & \Sigma \cup \{C < (\mathsf{all}\ f\ C_1)\} \\ & \cup \{C_1 < C_2 : \Sigma \models C < (\mathsf{all}\ f\ C_2)\} \models C_1 < D.\end{aligned}$$

We say that a subsumption constraint $C < D$ is *simple* if it is of the form $C < C'$, $C <$ (all f C'), and $C <$ (fd C' : $\mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$); Lemmas 7 and 8 allow us to convert general logical implication problems to (sets of) problems where all subsumption constraints are simple. For each such problem $\Sigma \models \varphi$ we define a Datalog$_{nS}$ recognition problem $P_\Sigma \cup P_\varphi \models G_\varphi$ as follows:

$$\begin{array}{lll}
P_\Sigma = \{\ \mathrm{cl}(X, c_j, Y) \leftarrow \mathrm{cl}(X, c_i, Y) & \text{for all } C_i < C_j \in \Sigma & (1)\\
\quad \mathrm{cl}(f(X), c_j, Y) \leftarrow \mathrm{cl}(X, c_i, Y) & \text{for all } C_i < (\mathsf{all}\ f\ C_j) \in \Sigma & (2)\\
\quad \mathrm{cl}(X, Y, 1) \leftarrow \mathrm{eq}(X), \mathrm{cl}(X, Y, 2) & & (3)\\
\quad \mathrm{cl}(X, Y, 2) \leftarrow \mathrm{eq}(X), \mathrm{cl}(X, Y, 1) & & (4)\\
\quad \mathrm{eq}(f(X)) \leftarrow \mathrm{eq}(X) & \text{for all primitive attributes } f & (5)\\
\multicolumn{2}{l}{\quad \mathrm{eq}(\overline{\mathsf{Pf}}(X)) \leftarrow \mathrm{cl}(X, c_i, 1), \mathrm{cl}(X, c_j, 2), \mathrm{eq}(\overline{\mathsf{Pf}}_1(X)), \ldots, \mathrm{eq}(\overline{\mathsf{Pf}}_k(X))} & (6)\\
\multicolumn{2}{l}{\quad \mathrm{eq}(\overline{\mathsf{Pf}}(X)) \leftarrow \mathrm{cl}(X, c_i, 2), \mathrm{cl}(X, c_j, 1), \mathrm{eq}(\overline{\mathsf{Pf}}_1(X)), \ldots, \mathrm{eq}(\overline{\mathsf{Pf}}_k(X))} & (7)\\
\multicolumn{2}{r}{\text{for all } C_i < (\mathsf{fd}\ C_j : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}) \in \Sigma\ \}} &
\end{array}$$

The clauses stand for the inferences of inheritance (1), direct typing (2), typing inferred from equalities (3-4), propagation of equality by primitive attributes (5), and path FD inference (6-7), respectively. In addition we use a set of facts to represent the left-hand-side of the posed question[2]:

$$P_\varphi = \begin{cases} \{\mathrm{cl}(0, c_i, 1)\} & \text{for } C_i < C_j \\ \{\mathrm{cl}(0, c_i, 1)\} & \text{for } C_i < (\mathsf{all}\ f\ C_j) \\ \{\mathrm{cl}(0, c_i, 1), \mathrm{cl}(0, c_j, 2), \mathrm{eq}(\overline{\mathsf{Pf}}_1(0)), \ldots, \mathrm{eq}(\overline{\mathsf{Pf}}_k(0))\} & \\ & \text{for } C_i < (\mathsf{fd}\ C_j : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}) \end{cases}$$

and a ground atom to represent the right-hand-side of the posed question:

$$G_\varphi = \begin{cases} \mathrm{cl}(0, c_j, 1) & \text{for } C_i < C_j \\ \mathrm{cl}(f(0), c_j, 1) & \text{for } C_i < (\mathsf{all}\ f\ C_j) \\ \mathrm{eq}(\overline{\mathsf{Pf}}(0)) & \text{for } C_i < (\mathsf{fd}\ C_j : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}) \end{cases}$$

Intuitively, the ground facts $\mathrm{cl}(\overline{\mathsf{Pf}}(0), c_j, i)$ and $\mathrm{eq}(\overline{\mathsf{Pf}}(0))$ derived from P_φ using P_Σ stand for properties of two distinguished nodes o_1 and o_2 and their descendents, in particular for $\mathsf{Pf}^I(o_i) \in C_j$ and $\mathsf{Pf}^I(o_1) = \mathsf{Pf}^I(o_2)$, respectively. In addition $|P_\Sigma| + |P_\varphi| \in \mathcal{O}(|\Sigma| + |\varphi|)$ and $|G_\varphi| \in \mathcal{O}(|\varphi|)$.

[2] Essentially an application of the Deduction theorem.

Theorem 9 *Let Σ be an arbitrary simple DLClass terminology and φ a simple DLClass subsumption constraint. Then* $\Sigma \models \varphi \iff P_\Sigma \cup P_\varphi \models G_\varphi$.

Proof: (sketch)

$\Leftarrow$: By induction on stages of $T_{P_\Sigma \cup P_\varphi}$ showing that every clause in $P_\Sigma \cup P_\varphi$ represents a valid inference (essentially the same as the "only-if" part of the proof of Theorem 4).

$\Rightarrow$: By contradiction we assume that $G_\varphi \notin T^\omega_{P_\Sigma \cup P_\varphi}$. We again construct an interpretation for DLClass starting with two complete infinite trees with edges labeled by primitive attribute names. We merge the two nodes accessible from the two distinct roots by the path Pf whenever $\mathrm{eq}(\overline{\mathsf{Pf}}(0)) \in T_{P_\Sigma \cup P_\varphi}$ (all children of such nodes are merged as well due to the clause $\mathrm{eq}(f(X)) \leftarrow \mathrm{eq}(X) \in P_\Sigma$). In addition we label each node n in the resulting graph by a set of class (identifier) labels c_i if $n = o_j.\,\mathsf{Pf}$ and $\mathrm{cl}(\overline{\mathsf{Pf}}(0), c_i, j) \in T_{P_\Sigma \cup P_\varphi}$ $(j = 1, 2)$.

Nodes of the resulting graph then provide the domain Δ of the interpretation; primitive concept C_i is interpreted as the set of nodes labeled c_i and the interpretation of primitive attributes is given by the edges of the graph.

The resulting interpretation satisfies Σ (by case analysis for the individual constraints in Σ using the corresponding clauses in P_Σ) and the "left-hand" side of φ (follows from the definition of P_φ), but falsifies the "right-hand" side of φ. $\Box$

This result completes the circle of reductions

$$\text{(normal) Datalog}_{nS} \longrightarrow \text{DLFD} \longrightarrow \text{DLClass} \longrightarrow \text{Datalog}_{nS}$$

Corollary 10 *Logical implication problems in DLFD and DLClass are DEXPTIME-complete.*

In particular, this also means that every DLClass problem can be reformulated as a DLFD problem, and consequently that *typing* and *inheritance* constraints do not truly enhance the expressive power of DLClass.

5 Polynomial Cases

Previous sections have established DEXPTIME-completeness for DLClass. However, there is an interesting syntactic restriction on uniqueness constraints in DLClass that (a) allows for a low-order polynomial time decision procedure that solves the logical implication problem, and (b) has a number of practical applications in the database area [16, 17]. In particular, the restriction requires all fd descriptions in a terminology Σ to be regular; in other words, to have one of the forms

1. $(\mathsf{fd}\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.\,\mathsf{Pf}', \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf})$ or
2. $(\mathsf{fd}\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}\,.\,\mathsf{Pf}', \ldots, \mathsf{Pf}_k \rightarrow \mathsf{Pf}\,.f)$.

Given the connection between DLFD and Datalog$_{nS}$ established by Theorem 4, a natural question is whether there is a syntactic restriction of Datalog$_{nS}$ programs that leads to an efficient decision procedure. We identify such a restriction in the following definition.

Definition 11 (Regular Datalog$_{nS}$) *Let P be a normal Datalog$_{nS}$ program. We say that P is* regular *if every clause with a non-empty body has the form*

$$p(t(X)) \leftarrow p_1(t_1(X)), \ldots, q(t'(t(X))), \ldots, p_k(t_k(X))$$

for some terms $t, t', t_1, \ldots, t_k$ (note that any of the terms may be just the variable X itself).

Theorem 12 *The recognition problem $P \models G$ for regular Datalog$_{nS}$ programs has a low-order polynomial time decision procedure.*

Proof: Consider the conversion of P to Σ_P presented in Theorem 4. It is not hard to see that such a conversion of any clause with non-empty body in P to a constraint in DLFD would obtain a regular fd description. The statement of the theorem then follows since the conversion of the recognition problem takes $\mathcal{O}(|P| + |G|)$ time, $|\Sigma_P| + |\varphi_{P,G}| \in \mathcal{O}(|P| + |G|)$, and the obtained (equivalent) logical implication problem can be solved using a $\mathcal{O}(|\Sigma_P| \cdot |\varphi_{P,G}|)$ procedure for regular DLClass problems [16]. □

In addition, a slight generalization of the regularity condition in DLClass leads to intractability [16]. The same turns out to be true for regular Datalog$_{nS}$.

Definition 13 (Nearly-regular Datalog$_{nS}$) *We define P as a* nearly regular *Datalog$_{nS}$ program if every clause with non-empty body has one of the forms*

1. $p(f(t(X))) \leftarrow p_1(t_1(X)), \ldots, q(t'(t(X))), \ldots, p_k(t_k(X))$ *or*
2. $p(t(f(X))) \leftarrow p_1(t_1(X)), \ldots, q(t'(t(X))), \ldots, p_k(t_k(X))$.

Essentially, near-regularity allows an additional function symbol f to appear in the head of a clause.

Theorem 14 *For an arbitrary normal Datalog$_{nS}$ program P there is an equivalent nearly regular Datalog$_{nS}$ program P'.*

Proof: Consider P' that contains the same facts as P, and for every clause

$$p(f_1(\ldots f_k(X)\ldots)) \leftarrow p_1(t_1(X)), \ldots, p_l(t_l(X)) \in P$$

with non-empty body, contains a set of clauses

$$\begin{array}{l} q_1(f_k(X)) \leftarrow p_1(t_1(X)), \ldots, p_l(t_l(X)), \\ q_2(f_{k-1}(f_k(X))) \leftarrow q_1(f_k(X)), \\ \qquad\vdots \\ q_k(f_1(\ldots f_k(X)\ldots)) \leftarrow q_{k-1}(f_2(\ldots f_k(X)\ldots)), \\ p(f_1(\ldots f_k(X)\ldots)) \leftarrow q_k(f_1(\ldots f_k(X)\ldots)) \end{array}$$

for some predicate symbols $q_1, \ldots, q_k$ not occurring in P. Each of the generated clauses is nearly regular (by satisfying the first form in Definition 13). Similarly, we could have generated P' where all clauses have satisfied the second form in Definition 13.

In both cases, the statement of the theorem is easy to establish from the construction of P'. □

Corollary 15 *The recognition problem for nearly regular Datalog$_{nS}$ programs is DEXPTIME-complete.*

6 Summary and Discussion

We have presented a description logic dialect called DLClass that is capable of capturing many important object relational database constraints including inheritance, typing, primary keys, foreign keys, and object identity. Although DLClass and its applications in information systems has been explored in earlier work [4, 16, 17], this paper establishes a strong connection with Datalog$_{nS}$. We have explored this connection and obtained the following results:

- An open problem relating to the complexity of DLFD (and therefore to the complexity of DLClass) that was originally considered in [15, 20] has been resolved. In particular, the result implies that the regularity condition for DLClass [16] is a boundary between tractable and intractable problems in DLClass.
- A consequence of our relatively straightforward linear reduction of DLClass to DLFD implies that inheritance and typing constraints do not appreciatively add to the utility of DLFD for capturing object relational schema. Such constraints are straightforwardly expressed in DLFD.
- We have identified a subset of the Datalog$_{nS}$ recognition problems that can be solved in PTIME. Moreover, we have shown that the regularity condition for Datalog$_{nS}$ programs is a boundary between polynomial and exponential time problems in Datalog$_{nS}$.

In general, a DL dialect is obtained by a selection of concept constructors that will in turn determine the tradeoff between the utility of the dialect for capturing schema structure and the complexity of the associated implication problem. For example, CLASSIC [3] opts for a less expressive selection that excludes negation and disjunction constructors in order to ensure an efficient polynomial time algorithm for its implication problem. Similarly, DLClass may be viewed as a minimal core of CLASSIC extended with uniqueness constraints. Conversely, other dialects opt for a more expressive selection that includes such constructors with the expectation that *model building* implication checking procedures work well on typical real world schema structure [10].

Note that DLCLass excludes any consideration for so-called *roles* which are essentially set-valued attributes. Allowing roles to be used in formulating uniqueness constraints is an interesting avenue for future work, although allowing them to be used only at the end of path expressions is a straightforward generalization of our results that would accommodate the notion of key in [9]. Also note that DLClass interprets attributes as total functions. However, partial functions can be straightforwardly simulated by employing a simple notational convention.

It is worth mentioning other work on various forms of path constraints for graph based data models relating to semi-structure data and object relational schema [5, 6, 13] that have similar objectives to research in the area of DLs. Indeed, many of the results in this work appear to have close analogies to corresponding work on DLs, although a more thorough exploration of this relationship is an interesting avenue for further research.

Another direction for future research relates to a property that DLFD shares with many DL dialects: its arbitrary and finite logical implication problems do not coincide. In particular,

$$\{\mathsf{THING} < (\mathsf{fd}\ \mathsf{THING} : A.B \to Id)\} \models \mathsf{THING} < (\mathsf{fd}\ \mathsf{THING} : B \to Id)$$

holds if and only if one requires the domain for any interpretation to be finite [15]. Furthermore, there is some evidence that existing techniques for finite model reasoning are not suited to more general information integration problems that involve access to large finite information sources such as the WEB [1].

Acknowledgements

The authors gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada and the Communications and Information Technology of Ontario for their support of this research. We also wish to thank the many helpful suggestions and comments by the anonymous referees.

References

1. Serge Abiteboul and Victor Vianu. Queries and computation on the web. In *6th International Conference on Database Theory ICDT'97*, volume 1186 of *Lecture Notes in Computer Science*, pages 262–275. Springer, 1997.
2. Alexander Borgida. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.
3. Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. Classic: A structural data model for objects. In James Clifford, Bruce G. Lindsay, and David Maier, editors, *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 58–67. ACM Press, 1989.
4. Alexander Borgida and Grant Weddell. Adding uniqueness constraints to description logics (preliminary report). In *International Conference on Deductive and Object-Oriented Databases*, pages 85–102, 1997.
5. Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured and structured databases. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 129–138, 1998.

6. Peter Buneman, Wenfei Fan, and Scott Weinstein. Interaction between path and type constraints. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 56–67, 1999.
7. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured Objects: modeling and reasonning. In *International Conference on Deductive and Object-Oriented Databases*, pages 229–246, 1995.
8. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering Queries Using Views in Description Logics. In *6th International Workshop on Knowledge Representation meets Databases (KRDB'99)*, pages 6–10, 1999.
9. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Keys for free in description logics. In *International Workshop on Description Logics DL2000*, pages 79–88, 2000.
10. Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modelling. In Jan Chomicki and Gunter Saake, editors, *Logics for Databases and Information Systems*, chapter 8. Kluwer, 1998.
11. Jan Chomicki. *Functional Deductive Databases: Query Processing in the Presence of Limited Function Symbols.* PhD thesis, Rutgers University, New Brunswick, New Jersey, January 1990. Also Laboratory for Computer Science Research Technical Report LCSR-TR-142.
12. Jan Chomicki and Tomasz Imieliński. Finite Representation of Infinite Query Answers. *ACM Transactions on Database Systems*, 18(2):181–223, June 1993.
13. Wenfei Fan and Jérôme Siméon. Integrity constraints for xml. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 23–34, 2000.
14. Martin Fürer. Alternation and the Ackermann Case of the Decision Problem. *L'Enseignement Math.*, 27:137–162, 1981.
15. Minoru Ito and Grant Weddell. Implication Problems for Functional Constraints on Databases Supporting Complex Objects. *Journal of Computer and System Sciences*, 49(3):726–768, 1994.
16. Vitaliy L. Khizder. *Uniqueness Constraints in Object-Relational Databases and Description Logics.* PhD thesis, University of Waterloo, 1999.
17. Vitaliy L. Khizder, David Toman, and Grant Weddell. Reasoning about Duplicate Elimination with Description Logic. Technical report, Dept. of Computer Science, University of Waterloo, 2000. (accepted to DOOD 2000).
18. John W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd edition, 1987.
19. Martin F. van Bommel and Grant Weddell. Reasoning About Equations and Functional Dependencies on Complex Objects. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):455–469, 1994.
20. Grant Weddell. A Theory of Functional Dependencies for Object Oriented Data Models. In *International Conference on Deductive and Object-Oriented Databases*, pages 165–184, 1989.
21. Grant Weddell. Reasoning about Functional Dependencies Generalized for Semantic Data Models. *ACM Transactions on Database Systems*, 17(1):32–64, 1992.