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
Email: {vlkhizde,david,gweddell}@uwaterloo.ca

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 (DL) have found many applications in information systems [3]. They also address problems in query optimization and information integration [1, 8, 9]. In such applications, it becomes essential to capture knowledge that relates to various kinds of uniqueness constraints. For example, in a hypothetical patient management system, it can be crucial to know that *a hospital has a unique name*, that *a patient is uniquely identified by hospital and patient number*, and that *a person admitted to a hospital has a valid unique social security number*. Concept constructors that enable capturing *keys* (in the standard database sense) and simple forms of *functional dependencies* in a DL have been proposed in [1, 2], a nagging "missing ingredient" for prior DL dialects that had efficient subsumption checking algorithms. For example the above constraints for hospitals and patients can be formalized by the following DL constraints [1]:

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

In this paper we consider a more general version of the fd constructor [7, 8, 9, 12], that allows component attribute descriptions to correspond to attribute

or feature *paths*. To focus on the essential idea, we define a DL dialect called DLFD consisting of a single concept constructor for this more general form of uniqueness constraint. The logical implication problem for DLFD is a special case of the logical implication problem for path functional dependencies (PFDs), a variety of uniqueness constraints for data models supporting complex objects first proposed in [12] and studied more fully in a subsequent series of papers [7, 11, 13].

Although DLFD is extremely simple, it has some surprising capabilities. In particular, logical implication problems in a more general DL dialect called DLClass (the examples above are legal constraints in DLClass) can be simulated in DLFD via a linear translation. DLClass includes other concept constructors essential to the convenient capture of object relational database schema in information systems, in particular class inheritance and attribute typing. The 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 [4, 5]. This relationship leads to the main contribution of the paper: resolving an open issue relating to the complexity of reasoning about PFDs. 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 [7] becomes tight.

The above example constraints in DLClass have a peculiar 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 [8]. 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 restriction 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 addresses special cases in which PTIME reasoning is possible. We conclude with a summary in Section 6.

2 Definitions

Definition 2.1 (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 descrip-

tions *using the following grammar:*

$$\begin{array}{rcl} D & ::= & C \\ & | & (\mathsf{all}\ f\ D) \\ & | & (\mathsf{fd}\ C : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}), k > 0 \\ & | & (\mathsf{and}\ D\ D) \end{array}$$

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,* $f^I : \Delta \to \Delta$. *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*

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

We say that an interpretation satisfies a constraint $C < D$ *if* $C^I \subseteq D^I$.
For a given set of subsumption constraints (a terminology) $\Sigma = \{C_i < D_i : 0 < i \leq n\}$ *and a subsumption constraint (a posed question)* $C < D$, *a* logical implication problem *is the question does* $\Sigma \models C < D$ *hold, i.e, do all interpretations that satisfy* Σ *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 no 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 2.2 (Datalog$_{nS}$ [5]) *Let* p_i *be predicate symbols,* f_i *function symbols such that* $p_i \neq f_i$, *and* $X, Y, \ldots$ *variables. A* logic program P *is a finite set of Horn clauses of the form*

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

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* $f(t, s_1, \ldots s_l)$ *where* f *a function symbol,* t *is a functional term, and* $s_1, \ldots, s_l$ *are data terms,*
2. s^i_j *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* $q(t, s_1, \ldots, s_k)$ *is the question does* $P \models q(t, s_1, \ldots, s_k)$ *hold, i.e.,* $q(t, s_1, \ldots, s_k)$ *is true in all models of* P*?*

It is known that every $Datalog_{nS}$ program can be encoded as a normal $Datalog_{nS}$ program [4, 5]. 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 [10]:

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; $M_P = T^\omega_P(\emptyset)$.

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

Proposition 2.3 ([5, 6]) *The recognition problem for* $Datalog_{nS}$ *programs is DEXPTIME-complete (under the data-complexity*[1] *measure). The lower bound holds even for programs in normal form.*

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.

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

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: with every path expression of the form $\mathsf{Pf} = f_1.\cdots.f_k.\mathit{Id}$ we associate two Datalog$_{nS}$ functional terms $\overline{\mathsf{Pf}}(0) = f_k(\cdots f_1(0)\cdots)$ and $\overline{\mathsf{Pf}}(X) = f_k(\cdots f_1(X)\cdots)$, 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 $\mathsf{Pf} = f_k.\cdots.f_1.\mathit{Id}$ 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 3.1 *Let P be an normal Datalog$_{nS}$ program and $G = p(\overline{\mathsf{Pf}}(0))$ a ground atom. We define*

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

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

Proof: $\Rightarrow$: We show that $\overline{\mathsf{Pf}}(0) \in T_P^\omega(\emptyset)$ 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 a 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 $\mathsf{Pf}'_1.p'_1, \ldots, \mathsf{Pf}'_l.p'_l \rightarrow \mathsf{Pf}'.p' \in \Sigma_P$. 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 respective paths from o_1 and o_2 to x_1 and x_2 (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. □

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

Corollary 3.2 *The logical implication problem for DLFD is DEXPTIME-hard.*

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

Corollary 3.3 *The logical implication problem for DLClass is DEXPTIME-hard.*

4 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$_{nS}$ [5]. We start with two lemmas that are used to simplify complex DLClass constraints.

Lemma 4.1 *Let C_1 be a primitive concept not in $\Sigma \cup \{C' < D'\}$. Then*

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

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

A terminology Σ is *simple* if it does not contain any descriptions of the form $(\mathsf{and}\ D_1\ D_2)$ and whenever $(\mathsf{all}\ f\ D)$ appears in Σ then D is a primitive concept description. Lemma 4.1 shows that every terminology can be converted to an equivalent simple terminology.

Lemma 4.2 *Let Σ be a simple terminology and C_1 a primitive concept not present in $\Sigma \cup \{C < D, C' < D'\}$. Then*

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

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

A subsumption constraint $C < D$ is *simple* if it is of the form $C < (\mathsf{all}\ f\ C')$, $C < C'$, and $C < (\mathsf{fd}\ C' : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf})$; Lemmas 4.1 and 4.2 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 } C_i < C_j \in \Sigma & (1)\\
\quad \mathrm{cl}(f(X), c_j, Y) \leftarrow \mathrm{cl}(X, c_i, Y) & \text{for } 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)\\
\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)\\
\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)\\
\qquad\qquad\qquad

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. $\square$

This result completes the circle of reductions

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

Corollary 4.4 *Membership 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 therefore the *typing* and *inheritance* constraints do not truly enhance the expressive power of DLClass.

5 Polynomial Cases

Previous sections have established DEXPTIME-completeness for logical implication in 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 [8, 9]. In particular, the restriction requires all fd descriptions in a terminology Σ to be regular; in other words, to have the form

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

Given the connection between DLFD and Datalog$_{nS}$ established by Theorem 3.1, 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 5.1 (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 5.2 *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 3.1. 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 [8]. □

In addition, a slight generalization of the regularity condition in DLClass leads to intractability [8]. The same turns out to be true for regular Datalog$_{nS}$.

Definition 5.3 (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 5.4 *For an arbitrary normal Datalog$_{nS}$ program P there is an equivalent nearly regular Datalog$_{nS}$ program P'.*

Corollary 5.5 *The recognition problem for nearly regular Datalog$_{nS}$ programs is DEXPTIME-complete.*

6 Conclusion

In summary, we have reviewed a description logic DLClass that is capable of capturing many important object relational database constraints including inheritance, typing, primary keys, foreign keys, and object identity. While this description logic, its extensions and their applications in the database world have been explored in earlier work [1, 8, 9], this paper establishes a strong connection between DLClass and Datalog$_{nS}$. We have explored the connection and obtained the following results:

(1) We have resolved an open problem relating to the complexity of DLFD (and therefore to the complexity of DLClass) that was originally considered in [7, 12]. In particular, the result implies that the regularity condition for DLClass [8] is a boundary between tractable and intractable problems in DLClass.

(2) A consequence of the equivalence of DLClass and DLFD implies that inheritance and typing constraints do not add expressive power beyond general uniqueness constraints.

(3) 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$_{nS}$.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Jan Chomicki and Tomasz Imieliński. Finite Representation of Infinite Query Answers. *ACM Transactions on Database Systems*, 18(2):181–223, June 1993.

[6] Martin Fürer. Alternation and the Ackermann Case of the Decision Problem. *L'Enseignement Math.*, 27:137–162, 1981.

[7] 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.

[8] Vitaliy L. Khizder. *Uniqueness Constraints in Object-Relational Databases and Description Logics.* PhD thesis, University of Waterloo, 1999.

[9] 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).

[10] John W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 2nd edition, 1987.

[11] 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.

[12] 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.

[13] Grant Weddell. Reasoning about Functional Dependencies Generalized for Semantic Data Models. *ACM Transactions on Database Systems*, 17(1):32–64, 1992.