# On Attributes, Roles, and Dependencies in Description Logics and the Ackermann Case of the Decision Problem

David Toman and Grant Weddell
Department of Computer Science, University of Waterloo
Email: {david,gweddell}@uwaterloo.ca

**Abstract**

We present a decision procedure for the logical implication problem of a boolean complete DL dialect that includes attributes, roles, inverse roles and a new concept constructor that is capable of expressing a variety of equality and order-generating dependencies. The procedure underlies a mapping of such problems to satisfiability problems of $\text{Datalog}^{\vee}_{nS}$ and in turn to the Ackermann case of the decision problem.

# 1 Introduction

Apart from *concepts*, which are descriptions of sets, *description logics* (DLs) that derive from $\mathcal{ALC}$ [24], such as $\mathcal{SHIQ}$ [17] and $\mathcal{DLR}$ [7], start with descriptions of relations as basic building blocks. Binary relations are a particular focus and are called *roles*. In this paper we take the alternative approach: we study a DL dialect, $\mathcal{DLF}$, that starts by augmenting concepts with the notion of *attributes* (or *features*). Attributes correspond to unary functions on an underlying object domain. We also study various extensions of $\mathcal{DLF}$, in particular the dialect $\mathcal{DLFR}$ that reintroduces roles and $\mathcal{DLFD}$ that introduces *equality- and order-generating dependencies* [1, 3]. This approach has several advantages. First, the decidability and complexity of logical implication for $\mathcal{DLF}$ and its variants are closely related to reasoning in $\text{Datalog}^{\vee}_{nS}$, a *logic programming* language of monadic predicates and functions [9, 10], and in turn to the classical *Ackermann case of the Decision Problem* [2, 4]. This connection allows us to draw on powerful techniques developed for the classical decision problems. Second, the use of $\text{Datalog}^{\vee}_{nS}$ provides an uniform framework in which we can study various extensions of $\mathcal{DLF}$, in particular how roles can be simulated by attributes or how dependencies can be introduced via additional monadic predicates. And third, the connection to logic programming provides a rich set of *efficient implementation* techniques [12, 13].

## 1.1 Related Work, Motivation and Outline

In [20, 21] we considered a very simple DL dialect that included an "$\mathsf{fd}$" concept constructor for capturing path functional dependencies [26]. An earlier form of this constructor, in which components were restricted to individual attributes or to the identity function, was first proposed in [5]. The implication problem for this dialect was complete for DEXPTIME despite the fact that it did not allow defined concepts in terminologies. This result was obtained by comparing the problem to the recognition problem for $\text{Datalog}_{nS}$.

The ability to formulate implication problems using $\mathsf{fd}$ has a number of important applications in query optimization. In particular, the correctness of a number of query rewrite rules that relate to duplicate elimination [20], to sort operator elimination [25], and to a variety of other optimizations [23, 26] can be usefully abstracted as logical implication problems. Such rewrites are special cases of the general problem of query containment which has been solved by existing DL dialects for conjunctive query languages that assume set semantics [16]. Since an analogous general solution for bag or possible result semantics [11] seems unlikely at this time [8], we believe an ability to handle more limited classes of query containment problems that correspond to such rewrite rules [20] is a new and valuable application of DL reasoning.

The remainder of the paper is organized as follows. Section 2 defines the syntax and semantics for $\mathcal{DLF}$ and its extensions, and introduces the Ackermann decision problem and $\text{Datalog}^{\vee}_{nS}$. Section 3.1 considers $\mathcal{DLF}$ and shows that its logical membership problem is already complete for DEXPTIME. Section 3.2 considers $\mathcal{DLFR}$ which adds roles and their inverses. An interesting novelty is the simulation of roles via attributes. In Section 3.3, we consider $\mathcal{DLFD}$ which adds a more general version of $\mathsf{fd}$ that derives from the class of ordering dependencies considered in [15]. We conclude with a summary and a list of conjectures and open questions in Section 4.

# 2 Definitions

The syntax and semantics of $\mathcal{DLF}$, $\mathcal{DLFR}$, and $\mathcal{DLFD}$ dialects of description (or feature) logics are given by the following.

**Definition 1 (Syntax and Semantics of $\mathcal{DLF}$)**
*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) and $R$ descriptions of primitive binary roles ($P$) and their inverses ($P^{-1}$). We define derived* concept descriptions *by the grammar in Figure 1. (Checkmarks indicate the productions that form part of the grammar for each of the various dialects: $\mathcal{DLF}$, $\mathcal{DLFR}$, and $\mathcal{DLFD}$.) A concept formed by an application of the final production in the grammar is called an* ordering concept.

*An* inclusion dependency *$\mathcal{C}$ is an expression of the form $D \sqsubseteq E$.*

*The* semantics *of expressions is given with respect to a structure $(\Delta, \leq, \cdot^{\mathcal{I}})$, where*

| SYNTAX:$^\dagger$ $\mathcal{DLF}/\mathcal{DLFR}/\mathcal{DLFD}$ | | SEMANTICS: DEFN OF "$(\cdot)^{\mathcal{I}}$" |
|---|---|---|
| $D ::= C$ | $\checkmark\checkmark\checkmark$ | $(C)^{\mathcal{I}} \subset \Delta$ |
| $\mid D_1 \sqcap D_2$ | $\checkmark\checkmark\checkmark$ | $(D_1)^{\mathcal{I}} \cap (D_2)^{\mathcal{I}}$ |
| $\mid \forall f.D$ | $\checkmark\checkmark\checkmark$ | $\{x : (f)^{\mathcal{I}}(x) \in (D)^{\mathcal{I}}\}$ |
| $\mid \neg D$ | $\checkmark\checkmark\checkmark$ | $\Delta \setminus (D)^{\mathcal{I}}$ |
| $\mid \forall R.D$ | $\checkmark$ | $\{x : \forall y \in \Delta.(x,y) \in (R)^{\mathcal{I}} \Rightarrow y \in (D)^{\mathcal{I}}\}$ |
| $\mid \exists R.D$ | $\checkmark$ | $\{x : \exists y \in \Delta.(x,y) \in (R)^{\mathcal{I}} \wedge y \in (D)^{\mathcal{I}}\}$ |
| $E ::= D$ | $\checkmark\checkmark\checkmark$ | |
| $\mid E_1 \sqcap E_2$ | $\checkmark$ | $(E_1)^{\mathcal{I}} \cap (E_2)^{\mathcal{I}}$ |
| $\mid \forall f.E$ | $\checkmark$ | $\{x : (f)^{\mathcal{I}}(x) \in (E)^{\mathcal{I}}\}$ |
| $\mid D\{\,\mathsf{Pf}_1^{\sim_1}, ..., \mathsf{Pf}_k^{\sim_k}\,\} \to \mathsf{Pf}^{\sim}$ | $\checkmark$ | $\{x : \forall y \in (D)^{\mathcal{I}}.$ $\bigwedge_{i=1}^{k} (\mathsf{Pf}_i)^{\mathcal{I}}(x) \sim_i (\mathsf{Pf}_i)^{\mathcal{I}}(y) \Rightarrow (\mathsf{Pf})^{\mathcal{I}}(x) \sim (\mathsf{Pf})^{\mathcal{I}}(y)\}$ |

$^\dagger$ $k > 0$, $\sim \in \{<, \leq, =, \geq, >\}$

Figure 1: SYNTAX AND SEMANTICS OF $\mathcal{DLF}$ AND EXTENSIONS.

*$\Delta$ is a domain of "objects"; $\leq$ is a partial (or linear) order over $\Delta$; and $(.)^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concepts to be subsets of $\Delta$, primitive attributes to be total functions over $\Delta$, and primitive binary roles to be subsets of $\Delta \times \Delta$. The interpretation is extended to path expressions, $(Id)^{\mathcal{I}} = \lambda x.x$ and $(f.\,\mathsf{Pf})^{\mathcal{I}} = (\mathsf{Pf})^{\mathcal{I}} \circ (f)^{\mathcal{I}}$, reverse roles, $(P^{-1})^{\mathcal{I}} = \{(y,x) : (x,y) \in (P)^{\mathcal{I}}\}$, and derived concept descriptions, cf. Figure 1.*

*An interpretation* satisfies an inclusion dependency *$\mathcal{C} (\equiv D \sqsubseteq E)$ if $(D)^{\mathcal{I}} \subseteq (E)^{\mathcal{I}}$. A* terminology *$\mathcal{T}$ consists of a finite set of inclusion dependencies $\{\mathcal{C}_1, ..., \mathcal{C}_m\}$. The* logical implication problem *asks if $\mathcal{T} \models \mathcal{C}$ holds; that is, if all interpretations that satisfy each constraint in $\mathcal{T}$ must also satisfy $\mathcal{C}$ (the posed question).*

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. Finally, we write $(\Delta, \cdot^{\mathcal{I}})$ to denote a structure $(\Delta, \leq, \cdot^{\mathcal{I}})$ for which "$\leq$" is equality (or does not matter).

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

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

**Definition 3 (Datalog$^{\vee}_{nS}$ [10])** *A* Datalog$^{\vee}_{nS}$ program *is a finite set of clauses constructed from monadic predicates $P_i$ and their negations, unary function symbols $f_i$, constants, and a single variable $x$ using the usual well-formedness rules. A* satisfiability problem *for a Datalog$^{\vee}_{nS}$ program $\Pi$ is to determine if $\Pi$ has a model. A Datalog$_{nS}$ program is a finite set of* definite Horn *Datalog$^{\vee}_{nS}$ clauses.*

*A* recognition problem *for a* $\text{Datalog}_{nS}$ *program* $\Pi$ *and a ground atom* $q$ *is to determine if* $q$ *is true in all models of* $\Pi$ *(i.e., if* $\Pi \cup \{\neg q\}$ *is not satisfiable).*

The unary function symbols are drawn from the same alphabet $F$ used for primitive attributes. Thus the path descriptions in $\mathcal{DLF}$ and terms in $\text{Datalog}^{\vee}_{nS}$ correspond to each other.

**Notation 4** $\overline{\mathsf{Pf}}(t)$ denotes a $\text{Datalog}^{\vee}_{nS}$ term of the form "$f_k(\cdots f_1(t)\cdots)$". This notation corresponds to the $\mathsf{Pf} = f_1.\cdots.f_k$ notation for path functions.

It is known that a $\text{Datalog}^{\vee}_{nS}$ program has a model if and only if it has a Herbrand model [9, 10, 22]; this allows us to use a syntactic techniques for model construction. To establish the complexity bounds we use the following result for the satisfiability of Ackermann formulae and in turn for $\text{Datalog}^{\vee}_{nS}$ programs.

**Proposition 5 ([10, 14])** *The complexity of satisfiability for* $\text{Datalog}^{\vee}_{nS}$ *programs is DEXPTIME-complete. The lower bound holds for* $\text{Datalog}_{nS}$ *programs.*

For the remainder of the paper, we use general first-order syntax for $\text{Datalog}^{\vee}_{nS}$ formulas instead of the clausal syntax to improve readability. However, each of the $\text{Datalog}^{\vee}_{nS}$ programs in the paper can be presented in a clausal form.

# 3 Decision Procedures and Complexity Bounds

In this section we elaborate on the relationship between several dialects of $\mathcal{DLF}$ and $\text{Datalog}^{\vee}_{nS}$. The main underlying idea lies in relating $\mathcal{DLF}$ concept descriptions to monadic predicates and in simulating $\mathcal{DLF}$ *structural properties* as assertions in $\text{Datalog}^{\vee}_{nS}$.

## 3.1 $\mathcal{DLF}$: Lower Bounds and Decision Procedure

First we show that every $\text{Datalog}_{nS}$ recognition problem can be simulated by a $\mathcal{DLF}$ implication problem.

**Definition 6** *Let* $\Pi$ *be a* $\text{Datalog}_{nS}$ *program and* $P(\overline{\mathsf{Pf}}(0))$ *a ground atom (a goal). We construct an implication problem for* $\mathcal{DLF}$ *as follows:*

- *a terminology from clauses in* $\Pi$*:*
$$\mathcal{T}_\Pi = \{\ \forall \mathsf{Pf}_1 .Q_1 \sqcap \ldots \sqcap \forall \mathsf{Pf}_k .Q_k \sqsubseteq \forall \mathsf{Pf} .P : P(\overline{\mathsf{Pf}}(x)) \leftarrow Q_1(\overline{\mathsf{Pf}}_1(x)), \ldots, Q_k(\overline{\mathsf{Pf}}_k(x)) \in \Pi\ \}$$
- *the posed question from ground facts in* $Q_i(\overline{\mathsf{Pf}}_i(0)) \in \Pi$ *and the goal* $G = P(\overline{\mathsf{Pf}}(0))$*:*
$$\mathcal{C}_{\Pi,G} = \forall \mathsf{Pf}_1 .Q_1 \sqcap \ldots \sqcap \forall \mathsf{Pf}_k .Q_k \sqsubseteq \forall \mathsf{Pf} .P$$

*where, assuming* $\overline{\mathsf{Pf}}(t) = f_k(\cdots f_1(t)\cdots)$*,* $\forall \mathsf{Pf} .D$ *denotes* $\forall f_1.\ldots\forall f_k.D$*.*

**Theorem 7** *Let* $\Pi$ *be a* $\text{Datalog}_{nS}$ *program and* $G$ *a goal. Then* $\Pi \models G \iff \mathcal{T}_\Pi \models \mathcal{C}_{\Pi,G}$*.*

The lower complexity bound then follows from [10, 14].

**Corollary 8** *The implication problem for $\mathcal{DLF}$ is DEXPTIME hard; this remains true when negation is removed from $\mathcal{DLF}$.*

This complements the hardness result for path functional dependencies [21]: while that result was obtained in a class-free setting, here we need (multiple) class labels. Note that the complexity is inherently connected with the $\forall f.D$ construct. Without it, the problem reduces to propositional satisfiability which is NP-complete.

Now we consider the other direction: can various dialects of $\mathcal{DLF}$ be (naturally) simulated by Datalog$^{\vee}_{nS}$? The remainder of this section shows that this is indeed possible. Starting with $\mathcal{DLF}$, we show how to construct a Datalog$^{\vee}_{nS}$ satisfiability problem that is equivalent to a given $\mathcal{DLF}$ implication problem. In the simulation, $\mathcal{DLF}$'s concept descriptions $D$ are modeled by monadic predicates $P_D(x)$. The construction proceeds in two steps. First, the structural properties of $\mathcal{DLF}$ are encoded as Datalog$^{\vee}_{nS}$ assertions.

**Definition 9 ($\mathcal{DLF}$ Concept Formation Constraints)** *Let $D$, $D_1$, and $D_2$ range over concept descriptions and $f$ over attribute names. We define*

$$\Pi_{\mathcal{DLF}} = \left\{ \begin{array}{l} \forall x.(P_D(x) \vee P_{\neg D}(x)), \forall x.\neg(P_D(x) \wedge P_{\neg D}(x)) \\ \forall x.P_{D_1 \sqcap D_2}(x) \leftrightarrow (P_{D_1}(x) \wedge P_{D_2}(x)) \\ \forall x.P_{\forall f.D}(x) \leftrightarrow P_D(f(x)) \end{array} \right\}.$$

The set $\Pi_{\mathcal{DLF}}$ captures the structural relationships between $\mathcal{DLF}$ concepts. Although this set is infinite in general, the set of concepts appearing in a particular implication problem, $\mathcal{T} \models \mathcal{C}$, is finite. Hence, one can restrict the set of assertions $\Pi_{\mathcal{DLF}}$ to a finite subset $\Pi^{\mathcal{T},\mathcal{C}}_{\mathcal{DLF}}$ that contains only predicates that define concepts in $\mathcal{T} \cup \{\mathcal{C}\}$ (we omit the superscripts whenever clear from the context). To complete the translation of a $\mathcal{DLF}$ implication problem, what remains is the translation of the inclusion constraints.

**Definition 10** *Let $\mathcal{T}$ and $\mathcal{C} \equiv D \sqsubseteq E$ be a $\mathcal{DLF}$ terminology and an inclusion constraint, respectively. We define*

- $\Pi_{\mathcal{T}} = \{\forall x.P_D(x) \rightarrow P_E(x) : D \sqsubseteq E \in \mathcal{T}\}$ *and*
- $\Pi_{\mathcal{C}} = \{P_D(0), P_{\neg E}(0)\}$.

The two clauses $\Pi_{\mathcal{C}}$ represent the skolemized version of $\neg\forall x.P_D(x) \rightarrow P_E(x)$; 0 is the Skolem constant for $x$. As usual, a model "containing" $\Pi_{\mathcal{C}}$ is a counterexample for $\mathcal{C}$.

**Theorem 11** *Let $\mathcal{T}$ and $\mathcal{C}$ be a terminology and inclusion dependency in $\mathcal{DLF}$, respectively. Then $\mathcal{T} \models \mathcal{C} \iff \Pi_{\mathcal{DLF}} \cup \Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$ is not satisfiable .*

This result shows that $\mathcal{DLF}$ is essentially an alternative *variable-free* syntax for (monadic) Datalog$^{\vee}_{nS}$. Also, as a consequence we have:

**Corollary 12** *The implication problem for $\mathcal{DLF}$ is DEXPTIME-complete.*

### 3.2 Adding Roles: $\mathcal{DLFR}$

Traditionally, description logics allow *roles*—binary relations between concepts. However, while general Ackermann formulae allow arbitrary arity relations in their matrix, they still require the use of a *single* universal ($\forall$) quantifier in their prefix. This prevents a direct formulation of the $\forall R.D$ concept since two $\forall$'s are needed (then adding unary function symbols leads to undecidable theories [4]).

Therefore we use a less direct formulation by modeling roles in $\mathcal{DLFR}$ via attributes. The essential problem is that $\exists R.D_i$ concepts can force a single object to be related via a role $R$ to multiple objects satisfying different constraints $D_i$, that, in general, may be disjoint. However, as all concept descriptions are essentially monadic predicates, one can syntactically determine the maximal number of such objects needed for a given implication problem.

**Definition 13 ($\exists$-rank)** *Let $\mathcal{T} \models \mathcal{C}$ be a $\mathcal{DLFR}$ implication problem. The number of distinct occurrences of $\exists R.D$ in $\mathcal{T} \cup \{\mathcal{C}\}$ is denoted* $\mathrm{Rank}(\mathcal{T}, \mathcal{C})$.

Now let $\mathcal{T}$ and $\mathcal{C}$ be fixed, and $\rho$ and $\delta_0, \ldots, \delta_l$, where $l = \mathrm{Rank}(\mathcal{T}, \mathcal{C})$, be function symbols neither in $\mathcal{T}$ nor in $\mathcal{C}$. We model a role $R$ by a monadic predicate $P_R$. The fact that $(o, o') \in (R)^{\mathcal{I}}$, for $o, o' \in \Delta$, is captured by asserting $P_R(\delta_i(o))$ and $o' = \rho(\delta_i(o))$ for some $0 \le i \le l$. The new predicates $P_R$ (and function symbols $\delta_i$ and $\rho$) are constrained to simulate the behavior of the $\forall R.D$ and $\exists R.D$ concepts by the following assertions:

$$\Pi_{\mathcal{DLFR}} = \Pi_{\mathcal{DLF}} \cup \left\{ \begin{array}{lcl} \forall x.P_{\exists R.D}(x) & \leftrightarrow & \bigvee_{j=0}^{l} [P_R(\delta_j(x)) \land P_D(\rho(\delta_j(x)))] \\ \forall x.P_{\forall R.D}(x) & \leftrightarrow & \bigwedge_{j=0}^{l} [P_R(\delta_j(x)) \to P_D(\rho(\delta_j(x)))] \end{array} \right\}.$$

For any fixed implication problem of size $n$ the size of the assertions is $O(n^2)$.

**Theorem 14** *Let $\mathcal{T}$ and $\mathcal{C}$ be a $\mathcal{DLFR}$ terminology and inclusion dependency. Then $\mathcal{T} \models \mathcal{C} \iff \Pi_{\mathcal{DLFR}} \cup \Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$* is not satisfiable. *The later can be decided in DEXPTIME.*

Note the similarity with the *tree* models for $\mathcal{DLR}$: essentially, one can convert a dag induced by roles into a tree by replacing $(o_1, o_2) \in (R)^{\mathcal{I}}$ by $(o_1, o_2')$ and $(o_1', o_2)$ where $o_1$ and $o_1'$ ($o_2$ and $o_2'$, respectively) belong to the same concepts. Such an interpretation is still a model.

### 3.2.1 Role Inverses and Other Role Constructs

To model an inverse role $R^{-1}$, we need to modify the definitions for $P_{\forall R.D}$ and $P_{\exists R.D}$ to take account of the fact that if a parent of an object (i.e., the argument of the function simulating the role) is related to a $\forall R.D$ object via an inverse $R^{-1}$, then this object must be related to the parent via the original role $R$. Thus the parent must satisfy $D$ (the $\exists R.D$ argument is similar). These observations are captured by the assertions (similarly for $P_{\forall R^{-1}.D}$ and $P_{\exists R^{-1}.D}$):

$$\begin{array}{lcll} \forall x.P_{\exists R.D}(x) & \leftrightarrow & \bigvee_{j=0}^{l} [P_R(\delta_j(x)) \land P_D(\rho(\delta_j(x)))] & \text{for } x \neq \rho(\delta_i(y)) \\ \forall x.P_{\exists R.D}(\rho(\delta_i(x))) & \leftrightarrow & \bigvee_{j=0}^{l} [P_R(\delta_j(\rho(\delta_i(x)))) \land P_D(\rho(\delta_j(\rho(\delta_i(x)))))] & \\ & & \quad \lor [P_{R^{-1}}(\delta_i(x)) \land P_D(x)] & \text{otherwise} \\ \forall x.P_{\forall R.D}(x) & \leftrightarrow & \bigwedge_{j=0}^{l} [P_R(\delta_j(x)) \to P_D(\rho(\delta_j(x)))] & \text{for } x \neq \rho(\delta_i(y)) \\ \forall x.P_{\forall R.D}(\rho(\delta_i(x))) & \leftrightarrow & \bigwedge_{j=0}^{l} [P_R(\delta_j(\rho(\delta_i(x)))) \to P_D(\rho(\delta_j(\rho(\delta_i(x)))))] & \\ & & \quad \land [P_{R^{-1}}(\delta_i(x)) \to P_D(x)] & \text{otherwise} \end{array}$$

Note that the $x \neq \rho(\delta_i(y))$ condition can be eliminated by enumerating all terms that do not have the above form (i.e., do not start with $\rho\delta_i$). Such an enumeration is finite (still $O(n^2)$). Theorem 14 and its proof naturally extend to this setting. Other constructs that involve roles, such as *numerical restrictions*, *role hierarchies*, *role constructors*, *roles with arity* $\geq 2$, and their combinations, can be similarly captured by appropriate assertions over the monadic predicates that model roles as long as the DL dialect itself has a tree model property [7, 17].

### 3.3 Adding Dependencies: $\mathcal{DLFD}$

In this section, we consider adding *guarded equality- and order-generating dependencies* to $\mathcal{DLF}$; a similar approach can be also used for $\mathcal{DLFR}$ since roles do not interact with the dependencies. We simplify $\mathcal{DLFD}$ implication problems as follows:

**Definition 15** *A $\mathcal{DLFD}$ implication problem $\mathcal{T} \models \mathcal{C}$ is* simple *if each inclusion dependency in $\mathcal{T}$ is of the form $D_1 \sqsubseteq D_2$ (a* pure *dependency) or the form $D_1 \sqsubseteq D_2\{\,\mathsf{Pf}_1^{\sim_1}, \ldots, \mathsf{Pf}_k^{\sim_k}\,\} \rightarrow \mathsf{Pf}^{\sim}$ (an* order dependency*).*

An arbitrary implication problem in $\mathcal{DLFD}$ can be reduced to (possibly several) simple implication problems [19, 21]. For each simple implication problem $\mathcal{T} \models \mathcal{C}$, we define a satisfiability problem $\Pi_{\mathcal{DLFD}} \cup \Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$. There are two cases to consider depending on $\mathcal{C}$.

**Case 1:** $\mathcal{C}$ is pure. Here we show that a large class of dependencies does not interfere with the decision procedures from Sections 3.1 and 3.2. This class of dependencies is defined as follows:

**Definition 16 (Guarded Order Dependency)**
*An order dependency $D_1 \sqsubseteq D_2\{\,\mathsf{Pf}_1^{\sim_1}, \ldots, \mathsf{Pf}_k^{\sim_k}\,\} \rightarrow \mathsf{Pf}^{\sim}$ is* guarded *if it satisfies the following conditions (conditions 2 and 3 are needed only for linear order):*

1. *if $= \subseteq \sim_i$ for all $0 < i \leq k$ then also $= \subseteq \sim$,*
2. *if $< \subseteq \sim_i$ for all $0 < i \leq k$ then also $< \subseteq \sim$ and*
3. *if $> \subseteq \sim_i$ for all $0 < i \leq k$ then also $> \subseteq \sim$,*

*where $\subseteq$ denotes set inclusion of relations.*

This definition eliminates problematic cases involving order dependencies, in particular any order dependency $\mathcal{C}$ that is apriory unsatisfiable. For guarded order dependencies, we have:

**Lemma 17** *Let $\mathcal{T}'$ be the set of all pure $\mathcal{DLF}$ inclusion dependencies in a $\mathcal{DLFD}$ terminology $\mathcal{T}$. Then (1) if $\mathcal{T}'$ has a model it also has a term (Herbrand) model and (2) if $(\Delta, (.)^{\mathcal{I}})$ is a term model of $\mathcal{T}'$ then $(\Delta, \leq, (.)^{\mathcal{I}})$ is a model of $\mathcal{T}$ in which the $\leq$ relation is defined for every pair of terms $t_1(0)\,(= s_1(f(t(0))))$ and $t_2(0)\,(= s_2(g(t(0))))$ in $\Delta$ by*

$$t_1(0) \leq t_2(0) \iff \begin{cases} t_1(0) = t_2(0) & \text{for partial order} \\ t_1(0) = t_2(0) \text{ or } f < g \text{ lexicographically} & \text{for linear order.} \end{cases}$$

This case of the implication problem therefore reduces to the problem in Section 3.1 (or 3.2) since, by the above Lemma, the dependencies in $\mathcal{T}$ do not interfere with the decision procedure: a model for $\mathcal{T}'$ can always be extended to a model of $\mathcal{T}$.

**Case 2:** $\mathcal{C} = D_1 \sqsubseteq D_2\{\,\mathsf{Pf}_1^{\sim_1},\ldots,\mathsf{Pf}_k^{\sim_k}\,\} \to \mathsf{Pf}^{\sim}$. To falsify such an order dependency, *two* objects (one in $D_1$ and another in $D_2$) that satisfy the preconditions of the dependency but fail to satisfy the conclusion are needed. We therefore construct two copies of the interpretation for the pure $\mathcal{DLF}$ constraints in $\mathcal{T}$ similarly to [26, 18]. However, as Herbrand terms are essentially the same in the two copies, it is sufficient to distinguish them by renaming the predicate symbols [21]. In addition, we need to model the "rules" of order, equality, and their interaction with concept descriptions. The structural rules for $\mathcal{DLFD}$ are thus defined as follows:

$$\Pi_{\mathcal{DLFD}} = \Pi^L_{\mathcal{DLF}} \cup \Pi^R_{\mathcal{DLF}} \cup \left\{\begin{array}{l} \forall x.\neg(O^<(x) \wedge O^=(x)) \\ \forall x.\neg(O^=(x) \wedge O^>(x)) \\ \forall x.\neg(O^<(x) \wedge O^>(x)) \\ \forall x.O^\leq(x) \leftrightarrow O^<(x) \vee O^=(x) \\ \forall x.O^\geq(x) \leftrightarrow O^>(x) \vee O^=(x) \end{array}\right\} \cup \left\{\begin{array}{l} \forall x.O^=(x) \to O^=(f(x)) \\ \forall x.O^=(x) \to (C^L(x) \leftrightarrow C^R(x)) \end{array}\right\},$$

where $\Pi^L$ is the set of assertions $\Pi$ in which every predicate $P_D$ is renamed to $P^L_D$ (similarly for $\Pi^R$). For linear orders we add the rule of excluded middle, $\forall x.O^<(x) \vee O^=(x) \vee O^>(x)$.

**Definition 18** *Let $\mathcal{T} \models \mathcal{C}$ be a simple $\mathcal{DLFD}$ implication problem for which $\mathcal{C}$ has the form $D_1 \sqsubseteq D_2\{\,\mathsf{Pf}_1^{\sim_1},\ldots,\mathsf{Pf}_k^{\sim_k}\,\} \to \mathsf{Pf}^{\sim}$, let $\mathcal{T}'$ be all pure dependencies in $\mathcal{T}$, and let $\mathcal{T}'' = \mathcal{T} \setminus \mathcal{T}'$. We define*

$$\Pi_{\mathcal{T}} = \Pi^L_{\mathcal{T}'} \cup \Pi^R_{\mathcal{T}'} \cup \left\{\begin{array}{r} \forall x.P^L_{D_i}(x) \wedge P^R_{D_j}(x) \wedge O^{\sim_1}(\overline{\mathsf{Pf}}_1(x)) \wedge \ldots \wedge O^{\sim_k}(\overline{\mathsf{Pf}}_k(x)) \to O^{\sim}(\overline{\mathsf{Pf}}(x)) \\ \forall x.P^R_{D_i}(x) \wedge P^L_{D_j}(x) \wedge O^{\sim_1}(\overline{\mathsf{Pf}}_1(x)) \wedge \ldots \wedge O^{\sim_k}(\overline{\mathsf{Pf}}_k(x)) \to O^{\sim}(\overline{\mathsf{Pf}}(x)) \\ \textit{for } D_i \sqsubseteq D_2\{\,\mathsf{Pf}_1^{\sim_1},\ldots,\mathsf{Pf}_k^{\sim_k}\,\} \to \mathsf{Pf}^{\sim} \in \mathcal{T}'' \end{array}\right\}$$

*and*

$$\Pi_{\mathcal{C}} = \{P^L_{D_1}(0), P^R_{D_2}(0), O^{\sim_1}(\overline{\mathsf{Pf}}_1(0)),\ldots,O^{\sim_k}(\overline{\mathsf{Pf}}_k(0)), \neg O^{\sim}(\overline{\mathsf{Pf}}(0))\}.$$

**Theorem 19** *Let $\mathcal{T} \models \mathcal{C}$ be a simple $\mathcal{DLFD}$ implication problem in which $\mathcal{C}$ is an order dependency. Then $\mathcal{T} \models \mathcal{C}$ if and only if $\Pi_{\mathcal{DLFD}} \cup \Pi_{\mathcal{T}} \cup \Pi_{\mathcal{C}}$ is not satisfiable.*

# 4 Summary

In this paper we address the problem of limited expressiveness of earlier DL dialects with path constraints [18, 21, 26]. We present a decision procedure for $\mathcal{DLFD}$, a boolean complete DL dialect that is capable of expressing a variety of equality and order-based dependencies, and that can be extended with roles and their inverses. This new procedure is based on tying the decision procedures of $\mathcal{DLF}$-like DL dialects to reasoning in a logic programming language $\text{Datalog}^{\vee}_{nS}$ and in turn to the classical Ackermann ($\exists^*\forall\exists^*$) decision problem. The relationship established in this paper allows us to transfer proof techniques (e.g., techniques based on Herbrand models) and complexity results.

So how "close to the cliff or the valley" have we come? One of the remaining limitations of $\mathcal{DLFD}$ is the lack of an ability to define roles that are inverse attributes. This would represent a first opportunity for roles and ordering concepts to interact. Although the details are beyond the scope of this paper, preliminary investigations reveal that it is possible to adapt the undecidability result of [6] to show that $\mathcal{DLFD}$ extended with inverse attributes will render its implication problem undecidable. However, we would still like to add some capacity, perhaps limited, to express inverse attributes while ensuring decidability. Syntactic restrictions on ordering concepts along the lines considered in [6] merit particular consideration. Second, the consequences of granting full first-order status to ordering concepts are not clear. Indeed, the epistemological significance of either a negated $\mathsf{fd}$ or the disjunction of two $\mathsf{fd}$s is unclear. Third, relaxation of the guarded requirement for order dependencies merits further study. Finally, we plan to investigate more general decidable constraint theories as the basis for path dependencies [3].

Another direction of research leads towards tractable dialects. Again, preliminary investigations suggest that there is a polynomial time procedure for the implication problem for a fragment of $\mathcal{DLFD}$ that excludes negation, disallows defined concepts, and requires ordering concepts that occur in terminologies to satisfy a syntactic condition similar to the *regularity* condition in [21].

# References


[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] Wilhelm Ackermann. *Solvable Cases of the Decision Problem.* Studies in Logic and the Foundations of Mathematics. North-Holland, 1954.

[3] Marainne Baudinet, Jan Chomicki, and Pierre Wolper. Constraint-Generating Dependencies. In *International Conference on Database Theory*, 1995.

[4] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer-Verlag, 1997.

[5] Alexander Borgida and Grant E. Weddell. Adding uniqueness constraints to description logics. In *International Conference on Deductive and Object Oriented Databases, DOOD*, pages 85–102, 1997.

[6] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification Constraints and Functional Dependencies in Description Logics. In *To be presented at 2001 IJCAI*, 2001.

[7] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description Logic Framework for Information Integration. In *Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[8] Surajit Chaudhuri and Moshe Y. Vardi. Optimization of *real* conjunctive queries. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 59–70, 1993.

[9] Jan Chomicki. *Functional Deductive Databases: Query Processing in the Presence of Limited Function Symbols.* PhD thesis, Rutgers University, 1990. Laboratory for Computer Science Research LCSR-TR-142.

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

[11] Neil Coburn and Grant E. Weddell. A Logic for Rule-Based Query Optimization in Graph-Based Data Models. In *International Conference on Deductive and Object Oriented Databases (DOOD)*, pages 120–145, 1993.

[12] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Declarative Problem-solving using the `dlv` System. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, 2000.

[13] Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. The KR System `dlv`: Progress Report, Comparisons and Benchmarks. In *Principles of Knowledge Representation and Reasoning, KR'98*, pages 406–417, 1998.

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

[15] Seymour Ginsburg and Richard Hull. Order Dependency in the Relational Model. *TCS*, 26:149–195, 1983.

[16] Ian Horrocks, Ulrike Sattler, Sergio Tessaris, and Stephan Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Workshop on Knowledge Representation meets Databases (KRDB 2000)*, number 29 in CEUR Workshop Proceedings, pages 59–72, 2000.

[17] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical Reasoning for Expressive Description Logics. In *Logic Programming and Automated Reasoning, LPAR'99*, pages 161–180, 1999.

[18] 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.

[19] Vitaliy L. Khizder, David Toman, and Grant Weddell. On Decidability and Complexity of Description Logics with Uniquwness Constraints. In *Description Logics 2000*, pages 193–202, 2000.

[20] Vitaliy L. Khizder, David Toman, and Grant Weddell. Reasoning about Duplicate Elimination with Description Logic. In *Rules and and Objects in Databases, DOOD 2000 (part of Computational Logic 2000)*, pages 1017–1032, 2000.

[21] Vitaliy L. Khizder, David Toman, and Grant E. Weddell. On Decidability and Complexity of Description Logics with Uniqueness Constraints. In *International Conference on Database Theory ICDT'01*, pages 54–67, 2001.

[22] John W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[23] Alberto O. Mendelzon and Peter T. Wood. Functional Dependencies in Horn Clause Queries. *TODS*, 16(1):31–55, 1991.

[24] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[25] David E. Simmen, Eugene J. Shekita, and Timothy Malkemus. Fundamental Techniques for Order Optimization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 57–67, 1996.

[26] Grant E. Weddell. Reasoning about Functional Dependencies Generalized for Semantic Data Models. *TODS*, 17(1):32–64, 1992.