A Decidable Very Expressive *n*-ary Description Logic for Database Applications (extended abstract)

Alessandro Artale, Enrico Franconi, Rafael Peñaloza, Francesco Sportelli

KRDB Research Centre, Free University of Bozen-Bolzano, Italy {artale, franconi, penaloza, sportelli}@inf.unibz.it

Abstract. We introduce DLR^+ , an extension of the *n*-ary propositionally closed description logic DLR to deal with attribute-labelled tuples (generalising the positional notation), projections of relations, and global and local objectification of relations, able to express inclusion, functional, key, and external uniqueness dependencies. The logic is equipped with both TBox and ABox axioms forming a DLR^+ knowledge base (KB). We show how a simple syntactic restriction on the appearance of projections sharing common attributes in the KB makes reasoning in the language decidable with the same computational complexity as DLR. The obtained DLR^{\pm} *n*-ary description logic is able to encode more thoroughly conceptual data models such as EER, UML, and ORM.

1 Introduction

We introduce the new description logic (DL) DLR^+ extending the *n*-ary DL DLR [Calvanese *et al.*, 2008], in order to capture more database oriented constraints. While DLR is an expressive logic, it lacks a number of expressive means that can be added without increasing the complexity of reasoning—when used in a carefully controlled way. The added expressivity is motivated by the increasingly use of DLs as an abstract conceptual layer over relational databases, both to reason over such conceptual models during the database design phase, and to answer ontology-mediated queries over databases.

A DLR TBox can express axioms involving (i) propositional combinations of concepts and (compatible) *n*-ary relations, (ii) concepts as unary projections of *n*-ary relations, and (iii) relations with a component selected to be of a certain type. For example, if Pilot and RacingCar are concepts and DrivesCar, DrivesMotobike, DrivesVehicle are binary relations, we can write the following axioms:

 $Pilot \sqsubseteq \exists [1] \sigma_{2:RacingCar} DrivesCar$

 $DrivesCar \sqcup DrivesMotobike \sqsubseteq DrivesVehicle$.

 \mathcal{DLR}^+ extends \mathcal{DLR} in the following way.

- While \mathcal{DLR} instances of *n*-ary relations are *n*-tuples of objects—whose components are identified by their position in the tuple—instances of relations in \mathcal{DLR}^+ are *attribute-labelled tuples*, i.e., tuples where each component is identified by an attribute and not by its position in the tuple (see, e.g., [Kanellakis, 1990]). For example, the relation Employee may have the signature:

Employee(firstname, lastname, dept, deptAddr) ,

and an instance of Employee could be the tuple:

- Attributes can be *renamed*, also with the goal to recover the positional perspectives on relations: firstname, lastname, dept, deptAddr $\geq 1, 2, 3, 4$.

 Relation projections allow to form new relations by projecting a given relation on some of its attributes. For example, if Person is a relation with signature Person (name, surname), it could be related to Employee as follows:

 $\exists [\texttt{firstname}, \texttt{lastname}] \texttt{Employee} \sqsubseteq \texttt{Person}$

firstname, lastname \rightleftharpoons name, surname.

- The objectification of a relation (also known as reification) is a concept whose instances are unique identifiers of the tuples instantiating the relation. Those identifiers could be unique only within an objectified relation (*local objectification*), or they could be uniquely identifying tuples independently on the relation they are instance of (*global objectification*). For example, the concept EmployeeC could be the global objectification of the relation Employee, assuming that there is a global one-to-one correspondence between pairs of values of the attributes firstname, lastname and EmployeeC instances:
 - $EmployeeC \equiv \bigcirc \exists [firstname, lastname] Employee.$

As an example of local objectification, let us consider the two (ternary) relations OwnsCar (name, surname, car) and DrivesCar (name, surname, car), and assume that anybody driving a car also owns it; that is, DrivesCar \sqsubseteq OwnsCar. The locally objectified events of driving a car and owning a car, defined by the axioms CarDrivingEvent $\equiv \bigcirc$ DrivesCar and CarOwningEvent $\equiv \bigcirc$ OwnsCar, model the fact that a driving event by a person of a car is not necessarily the owning event by the same person and the same car. Indeed, they should be disjoint: CarDrivingEvent \Box CarOwningEvent $\sqsubseteq \bot$ should hold.

It turns out that \mathcal{DLR}^+ is an expressive description logic able to assert relevant constraints in the context of relational databases. In Section 3 we consider *inclusion dependencies, functional and key dependencies, external uniqueness* and *identification* axioms. For example, \mathcal{DLR}^+ can express the fact that the attributes firstname, lastname play the role of a multi-attribute key for the relation Employee:

 $\exists [\texttt{firstname}, \texttt{lastname}] \texttt{Employee} \sqsubseteq \exists \leq 1 [\texttt{firstname}, \texttt{lastname}] \texttt{Employee}, and that the attribute deptAddr functionally depends on the attribute dept within the relation \texttt{Employee}:}$

 $\exists [dept] Employee \sqsubseteq \exists \leq 1 [dept] (\exists [dept, deptAddr] Employee)$.

While \mathcal{DLR}^+ turns out to be undecidable, in this paper we show how a simple syntactic condition on the appearance of projections sharing common attributes in the knowledge base makes the language decidable. The result of this restriction is a new language called \mathcal{DLR}^{\pm} . We prove that \mathcal{DLR}^{\pm} , while preserving most of the \mathcal{DLR}^+ expressivity, has a reasoning problem whose complexity does not increase w.r.t. the computational complexity of the basic \mathcal{DLR} language. We also present in Section 5 the implementation of an API for the reasoning services in \mathcal{DLR}^{\pm} .

2 The Description Logic \mathcal{DLR}^+

A \mathcal{DLR}^+ signature is a tuple $\mathcal{L} = (\mathcal{C}, \mathcal{R}, \mathcal{O}, \mathcal{U}, \tau)$ where $\mathcal{C}, \mathcal{R}, \mathcal{O}$ and \mathcal{U} are finite, mutually disjoint sets of *concept names*, *relation names*, *individual names*, and *attributes*, respectively, and τ is a *relation signature* function, associating a set of attributes to each relation name $\tau(RN) = \{U_1, \ldots, U_n\} \subseteq \mathcal{U}$ with $n \ge 2$.

The syntax of concepts C, relations R, formulas φ , and attribute renaming axioms ϑ is given in Figure 1, where $CN \in C$, $RN \in \mathcal{R}$, $U \in \mathcal{U}$, $o \in \mathcal{O}$, q is a positive integer and $2 \leq k < \text{ARITY}(R)$. The *arity* of a relation R is the number of the attributes in its signature; i.e., $\text{ARITY}(R) = |\tau(R)|$, where we extend the signature function τ to arbitrary relations

Fig. 1. Syntax of \mathcal{DLR}^+ .

$$\begin{split} \tau(R_1 \backslash R_2) &= \tau(R_1) & \text{if } \tau(R_1) = \tau(R_2) \\ \tau(R_1 \sqcap R_2) &= \tau(R_1) & \text{if } \tau(R_1) = \tau(R_2) \\ \tau(R_1 \sqcup R_2) &= \tau(R_1) & \text{if } \tau(R_1) = \tau(R_2) \\ \tau(\sigma_{U_i:C}R) &= \tau(R) & \text{if } U_i \in \tau(R) \\ \tau(\exists^{\lessgtr q}[U_1, \dots, U_k]R) &= \{U_1, \dots, U_k\} & \text{if } \{U_1, \dots, U_k\} \subset \tau(R) \\ \tau(R) &= \varnothing & \text{otherwise} \end{split}$$

Fig. 2. The signature of \mathcal{DLR}^+ relations.

as specified in Figure 2. Notice that, while global objectification ($\bigcirc R$) can be applied to arbitrary relations, local ones ($\bigcirc RN$) can be applied just to relation names. We use the shortcut $\exists [U_1, \ldots, U_k] R$ for $\exists^{\geq 1} [U_1, \ldots, U_k] R$ for $k \geq 1$.

A \mathcal{DLR}^+ *TBox* \mathcal{T} is a finite set of *concept inclusion* axioms of the form $C_1 \subseteq C_2$ and *relation inclusion* axioms of the form $R_1 \subseteq R_2$. We use $X_1 \equiv X_2$ as a shortcut for $X_1 \subseteq X_2$ and $X_2 \subseteq X_1$. A \mathcal{DLR}^+ *ABox* \mathcal{A} is a finite set of *concept instance* axioms of the form CN(o), *relation instance* axioms of the form $RN(U_1:o_1, \ldots, U_n:o_n)$, and *same/distinct individual* axioms of the form $o_1 = o_2$ and $o_1 \neq o_2$, with $o_i \in \mathcal{O}$. Restricting ABox axioms to concept and relation names only does not affect the expressivity of \mathcal{DLR}^+ due to the availability of TBox axioms.

A set of renaming axioms forms a *renaming schema*, inducing an equivalence relation $(\rightleftharpoons, \mathcal{U})$ over the attributes \mathcal{U} , providing a partition of \mathcal{U} into equivalence classes each one representing alternative ways to name attributes. We write $[U]_{\Re}$ to denote the equivalence class of the attribute U w.r.t. the equivalence relation $(\rightleftharpoons, \mathcal{U})$. We allow only *well founded* renaming schemas, i.e., there is no equivalence class containing two attributes from the same relation signature. We use the shortcut $U_1 \dots U_n \rightleftharpoons U'_1 \dots U'_n$ to group many renaming axioms with the meaning that $U_i \rightleftharpoons U'_i$ for all $i = 1, \dots, n$.

The renaming schema reconciles the named attribute and the positional perspectives on relations. They are crucial when expressing both inclusion axioms and set operators $(\Box, \sqcup, \backslash)$ between relations, which make sense only over *union compatible* relations. Two relations R_1, R_2 are union compatible if their signatures are equal up to the attribute renaming induced by the renaming schema \Re ; that is, $\tau(R_1) = \{U_1, \ldots, U_n\}$ and $\tau(R_2) = \{V_1, \ldots, V_n\}$ have the same arity n and $[U_i]_{\Re} = [V_i]_{\Re}$ for each $1 \le i \le n$. Notice that through the renaming schema relations can use just local attribute names that can then be renamed when composing relations. Also note that it is obviously possible for the same attribute to appear in the signature of different relations.

A \mathcal{DLR}^+ knowledge base (KB) $\mathcal{KB} = (\mathcal{T}, \mathcal{A}, \Re)$ is composed by a TBox \mathcal{T} , an ABox \mathcal{A} , and a renaming schema \Re .

Example 1. Consider the relation names R_1, R_2 where $\tau(R_1) = \{W_1, W_2, W_3, W_4\}, \tau(R_2) = \{V_1, V_2, V_3, V_4, V_5\}$, and the renaming axiom $W_1W_2W_3 \rightleftharpoons V_3V_4V_5$. The TBox \mathcal{T}_{exa} consists of the axioms:

$$\exists [W_1, W_2] R_1 \sqsubseteq \exists^{\leq 1} [W_1, W_2] R_1 \tag{1}$$

$$\exists [V_3, V_4] R_2 \sqsubseteq \exists^{\leq 1} [V_3, V_4] (\exists [V_3, V_4, V_5] R_2) \tag{2}$$

$$\begin{split} & \left| \begin{array}{c} \top^{\mathcal{I}} = \varDelta \\ \bot^{\mathcal{I}} = \varnothing \\ (\neg C)^{\mathcal{I}} = \top^{\mathcal{I}} \backslash C^{\mathcal{I}} \\ (C_{1} \sqcap C_{2})^{\mathcal{I}} = C_{1}^{\mathcal{I}} \cap C_{2}^{\mathcal{I}} \\ (C_{1} \sqcup C_{2})^{\mathcal{I}} = C_{1}^{\mathcal{I}} \cup C_{2}^{\mathcal{I}} \\ (C_{1} \sqcup C_{2})^{\mathcal{I}} = S_{1}^{\mathcal{I}} \cup C_{2}^{\mathcal{I}} \\ (\overline{\exists}^{\leq q}[U_{i}]R)^{\mathcal{I}} = \{d \in \varDelta \mid |\{t \in R^{\mathcal{I}} \mid t[\rho(U_{i})] = d\}| \leq q\} \\ (\overline{\odot} R)^{\mathcal{I}} = \{d \in \varDelta \mid d = \iota(t) \land t \in R^{\mathcal{I}}\} \\ (\overline{\odot} RN)^{\mathcal{I}} = \{d \in \varDelta \mid d = \ell_{RN}(t) \land t \in RN^{\mathcal{I}}\} \\ (R_{1} \backslash R_{2})^{\mathcal{I}} = R_{1}^{\mathcal{I}} \backslash R_{2}^{\mathcal{I}} \\ (R_{1} \sqcap R_{2})^{\mathcal{I}} = R_{1}^{\mathcal{I}} \cap R_{2}^{\mathcal{I}} \\ (R_{1} \sqcup R_{2})^{\mathcal{I}} = \{t \in R^{\mathcal{I}} \cup R_{2}^{\mathcal{I}} \mid \rho(\tau(R_{1})) = \rho(\tau(R_{2}))\} \\ (\sigma_{U_{i}:C}R)^{\mathcal{I}} = \{t \in R^{\mathcal{I}} \mid t[\rho(U_{i})] \in C^{\mathcal{I}}\} \\ (\exists^{\leq q}[U_{1}, \ldots, U_{k}]R)^{\mathcal{I}} = \{\langle \rho(U_{1}) : d_{1}, \ldots, \rho(U_{k}) : d_{k} \rangle \in T_{\Delta}(\{\rho(U_{1}), \ldots, \rho(U_{k})\}) \mid \\ 1 \leq |\{t \in R^{\mathcal{I}} \mid t[\rho(U_{1})] = d_{1}, \ldots, t[\rho(U_{k})] = d_{k}\}| \leq q\} \end{split}$$

Fig. 3. Semantics of \mathcal{DLR}^+ expressions.

$$\exists [W_1, W_2, W_3] R_1 \sqsubseteq \exists [V_3, V_4, V_5] R_2.$$
(3)

Intuitively, the axiom (1) expresses that W_1, W_2 form a multi-attribute key for R_1 ; (2) introduces a functional dependency in the relation R_2 where the attribute V_5 is functionally dependent from attributes V_3, V_4 , and (3) states an inclusion between two projections of the relation names R_1, R_2 based on the renaming schema axiom.

The semantics of \mathcal{DLR}^+ is based on *labelled tuples* over a domain Δ : a \mathcal{U} -*labelled tuple over* Δ (or *tuple* for short) is a function $t: \mathcal{U} \to \Delta$. For $U \in \mathcal{U}$, we write t[U] to refer to the domain element $d \in \Delta$ labelled by U, if the function t is defined for U—that is, if the attribute U is a label of the tuple t. Given $d_1, \ldots, d_n \in \Delta$, the expression $\langle U_1: d_1, \ldots, U_n: d_n \rangle$ stands for the tuple t such that $t[U_i] = d_i$, for $1 \leq i \leq n$. The *projection* of the tuple t over the attributes U_1, \ldots, U_k (i.e., the function t restricted to be undefined for the labels not in U_1, \ldots, U_k) is denoted by $t[U_1, \ldots, U_k]$. The relation signature function τ can be applied also to labelled tuples to obtain the set of labels on which the tuple is defined. $T_{\Delta}(\mathcal{U})$ denotes the set of all \mathcal{U} -labelled tuples over Δ .

A \mathcal{DLR}^+ interpretation is a tuple $\mathcal{I} = (\Delta, \mathcal{I}, \rho, i, L)$ consisting of a nonempty domain Δ , an interpretation function \mathcal{I} , a renaming function ρ , a global objectification function i, and a family L containing one local objectification function ℓ_{RN_i} for each named relation $RN_i \in \mathcal{R}$.

The renaming function ρ is a total function $\rho: \mathcal{U} \to \mathcal{U}$ representing a canonical renaming for all attributes. We use $\rho(\{U_1, \ldots, U_k\})$ to denote $\{\rho(U_1), \ldots, \rho(U_k)\}$. The global objectification function is an injective function, $i: T_{\Delta}(\mathcal{U}) \to \Delta$, associating a *unique* global identifier to each tuple. The local objectification functions, $\ell_{RN_i}: T_{\Delta}(\mathcal{U}) \to \Delta$, are associated to each relation name in the signature, and as the global objectification function they are injective: they associate an identifier—which is guaranteed to be unique only within the interpretation of a relation name—to each tuple. The interpretation function $\cdot^{\mathcal{I}}$ assigns a domain element to each individual $o^{\mathcal{I}} \in \Delta$, a set of domain elements to each concept name $CN^{\mathcal{I}} \subseteq \Delta$, and a set of \mathcal{U} -labelled tuples over Δ to each relation name conforming with its signature and to the renaming function $RN^{\mathcal{I}} \subseteq T_{\Delta}(\{\rho(U) \mid U \in \tau(RN)\})$. Note that the unique name assumption is not enforced. The interpretation function $\cdot^{\mathcal{I}}$ is unambiguously extended over concept and relation expressions as specified in Figure 3.

The interpretation \mathcal{I} satisfies the concept inclusion axiom $C_1 \subseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, and the relation inclusion axiom $R_1 \subseteq R_2$ if $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$. It satisfies the concept instance axiom CN(o) if $o^{\mathcal{I}} \in CN^{\mathcal{I}}$, the relation instance axiom $RN(U_1:o_1, \ldots, U_n:o_n)$ if $\langle \rho(U_1): o_1^{\mathcal{I}}, \ldots, \rho(U_n): o_n^{\mathcal{I}} \rangle \in RN^{\mathcal{I}}$, and the axioms $o_1 = o_2$ and $o_1 \neq o_2$ if $o_1^{\mathcal{I}} = o_2^{\mathcal{I}}$, and $o_1^{\mathcal{I}} \neq o_2^{\mathcal{I}}$, respectively. \mathcal{I} satisfies a renaming schema \Re if for every $U, V \in \mathcal{U}$, (i) $\rho(U) \in [U]_{\Re}$, and (ii) if $V \in [U]_{\Re}$, then $\rho(U) = \rho(V)$. \mathcal{I} is a *model* of the KB $(\mathcal{T}, \mathcal{A}, \Re)$ if it satisfies the axioms in the TBox \mathcal{T} , in the ABox \mathcal{A} , and the renaming schema \Re .

KB satisfiability refers to the problem of deciding the existence of a model of a given KB; concept satisfiability (resp. relation satisfiability) is the problem of deciding whether there is a model of the KB with a non-empty interpretation of a given concept (resp. relation). A KB entails (or logically implies) an axiom if all models of the KB are also models of the axiom. For instance, the TBox in Example 1 entails that axiom (2) is redundant since V_3 , V_4 are a key for R_2 : $\mathcal{T}_{\text{exa}} \models \exists [V_3, V_4] R_2 \sqsubseteq \exists^{\leq 1} [V_3, V_4] R_2$.

3 Expressiveness of \mathcal{DLR}^+

 \mathcal{DLR}^+ is an expressive description logic able to assert relevant constraints in the context of relational databases, such as *inclusion dependencies* (inclusion axioms among arbitrary projections of relations), *equijoins, functional dependency* axioms, *key* axioms, *external uniqueness* axioms, *identification* axioms, and *path functional dependencies*.

An *equijoin* among two relations with disjoint signatures is the set of all combinations of tuples in the relations that are equal on their selected attribute names. Let R_1, R_2 be relations with $\tau(R_1) = \{U, U_1, \ldots, U_{n_1}\}$ and $\tau(R_2) = \{V, V_1, \ldots, V_{n_2}\}$; their equijoin over U and V is the relation $R = R_1 \bowtie_{U=V} R_2$ that uses the signature

 $\tau(R) = \tau(R_1) \cup \tau(R_2) \setminus \{V\}, \text{ and is expressed by the } \mathcal{DLR}^+ \text{ axioms:} \\ \exists [U, U_1, \dots, U_{n_1}] R \equiv \sigma_{U: (\exists [U] R_1 \sqcap \exists [V] R_2)} R_1 \\ \exists [V, V_1, \dots, V_{n_2}] R \equiv \sigma_{V: (\exists [U] R_1 \sqcap \exists [V] R_2)} R_2 \\ U \rightleftharpoons V.$

A functional dependency axiom $(R: U_1 \dots U_j \to U)$ (also called *internal uniqueness* axiom [Halpin and Morgan, 2008]) states that the values of the attributes $U_1 \dots U_j$ uniquely determine the value of the attribute U in the relation R. Formally, the interpretation \mathcal{I} satisfies this functional dependency axiom if, for all tuples $s, t \in R^{\mathcal{I}}$, $s[U_1] = t[U_1], \dots, s[U_j] = t[U_j]$ imply s[U] = t[U]. Functional dependencies can be expressed in \mathcal{DLR}^+ , assuming that $\{U_1, \dots, U_j, U\} \subseteq \tau(R)$, with the axiom:

$$\exists [U_1, \dots, U_j] R \sqsubseteq \exists^{\leq 1} [U_1, \dots, U_j] (\exists [U_1, \dots, U_j, U] R)$$

A special case of functional dependencies are key axioms $(R: U_1 \dots U_j \rightarrow R)$, which state that the values of the key attributes $U_1 \dots U_j$ of a relation R uniquely identify tuples in R. A key axiom can be expressed in \mathcal{DLR}^+ , assuming that $\{U_1 \dots U_j\} \subseteq \tau(R)$, with the axiom:

 $\exists [U_1, \ldots, U_j] R \sqsubseteq \exists^{\leq 1} [U_1, \ldots, U_j] R.$ The *external uniqueness* axiom $([U^1] R_1 \downarrow \ldots \downarrow [U^h] R_h)$ states that the join R of the relations R_1, \ldots, R_h via the attributes U^1, \ldots, U^h has the joined attribute functionally dependent on all the others [Halpin and Morgan, 2008]. This can be expressed in \mathcal{DLR}^+ with the axioms:

 $\mathcal{DLR}^+ \text{ with the axioms:} \\ R \equiv R_1 \bowtie_{U^1 = U^2} \cdots \bowtie_{U^{h-1} = U^h} R_h \\ R : U_1^1, \dots, U_{n_1}^1, \dots, U_1^h, \dots, U_{n_h}^h \to U^1$

where $\tau(R_i) = \{U^i, U^i_1, \dots, U^i_{n_i}\}, 1 \le i \le h$, and R a new relation name with signature $\tau(R) = \{U^1, U^1_1, \dots, U^1_{n_1}, \dots, U^h_{n_1}, \dots, U^h_{n_h}\}.$

Identification axioms as defined in \mathcal{DLR}_{ifd} [Calvanese et al., 2001] (an extension of \mathcal{DLR} with functional dependencies and identification axioms) are a variant of external uniqueness axioms, constraining only the elements of a concept C; they can be expressed in \mathcal{DLR}^+ with the axiom:

 $[U^1]\sigma_{U_1:C}R_1\downarrow\ldots\downarrow[U^h]\sigma_{U_h:C}R_h$. Path functional dependencies—as defined in the DL family CFD [Toman and Weddell, 2009]—can be expressed in \mathcal{DLR}^+ as identification axioms involving joined sequences of functional binary relations. \mathcal{DLR}^+ also captures tree-based identification constraint (tid) introduced in [Calvanese et al., 2014] to capture fds in DL-Lite_{RDFS,tid}.

The rich set of constructors in \mathcal{DLR}^+ allows us to extend the known mappings in description logics of popular conceptual data models. The EER mapping as introduced in [Artale et al., 2007] can be extended to deal with multi-attribute keys (by using identification axioms) and named roles in relations; the ORM mapping as introduced in [Franconi et al., 2012; Sportelli and Franconi, 2016] can be extended to deal with arbitrary subset and exclusive relation constructs (by using inclusions among global objectifications of projections of relations), arbitrary internal and external uniqueness constraints, arbitrary frequency constraints (by using projections), local objectification, named roles in relations, and fact type readings (by using renaming axioms); the UML mapping as introduced in [Berardi et al., 2005] can be fixed to deal properly with association classes (by using local objectification) and named roles in associations.

The \mathcal{DLR}^{\pm} fragment of \mathcal{DLR}^{+} 4

Since a \mathcal{DLR}^+ KB can express inclusions and functional dependencies, the reasoning is undecidable [Chandra and Vardi, 1985]. In this section we present \mathcal{DLR}^{\pm} , a decidable syntactic fragment of \mathcal{DLR}^+ limiting the co-occurence of relation projections in a KB.

Given a \mathcal{DLR}^+ knowledge base $(\mathcal{T}, \mathcal{A}, \Re)$, the projection signature is the set \mathscr{T} containing the signatures $\tau(RN)$ of the relations $RN \in \mathcal{R}$, the singleton sets associated with each attribute name $U \in \mathcal{U}$, and the relation signatures that appear explicitly in projection constructs in some axiom from \mathcal{T} , together with their implicit occurrences due to the renaming schema. Formally, \mathscr{T} is the smallest set where (i) $\tau(RN) \in \mathscr{T}$ for all $RN \in \mathcal{R}$; (ii) $\{U\} \in \mathcal{T}$ for all $U \in \mathcal{U}$; and (iii) $\{U_1, \ldots, U_k\} \in \mathcal{T}$ for all $\exists \leq q [V_1, \ldots, V_k] R$ appearing as sub-formulas in \mathcal{T} and $\{U_i, V_i\} \subseteq [U_i]_{\Re}$ for $1 \leq i \leq k$.

The projection signature graph is the directed acyclic graph (\supset, \mathscr{T}) whose sinks are the attribute singletons $\{U\}$. Given a set of attributes $\tau = \{U_1, \ldots, U_k\} \subseteq \mathcal{U}$, the projection signature graph dominated by τ , denoted as \mathscr{T}_{τ} , is the sub-graph of (\supset, \mathscr{T}) containing all the nodes reachable from τ . Given two sets of attributes $\tau_1, \tau_2 \subseteq \mathcal{U}$, $\operatorname{PATH}_{\mathscr{T}}(\tau_1,\tau_2)$ denotes the set of paths in (\supset,\mathscr{T}) between τ_1 and τ_2 . Notice that PATH $\mathscr{T}(\tau_1, \tau_2) = \emptyset$ both when a path does not exist and when $\tau_1 \subseteq \tau_2$. The notation CHILD $\mathcal{T}(\tau_1, \tau_2)$ means that τ_2 is a child of τ_1 in (\supset, \mathcal{T}) . We now introduce \mathcal{DLR}^{\pm} .

Definition 1. A \mathcal{DLR}^{\pm} knowledge base is a \mathcal{DLR}^{+} knowledge base that satisfies the following syntactic conditions:

- 1. the projection signature graph (\supset, \mathscr{T}) is a multitree: i.e., for every node $\tau \in \mathscr{T}$, the graph \mathcal{T}_{τ} is a tree; and
- 2. for every projection construct $\exists^{\leq q}[U_1, \ldots, U_k]R$ appearing in \mathcal{T} , if q > 1 then the length of the path PATH $\mathcal{T}(\tau(R), \{U_1, \ldots, U_k\})$ is 1.

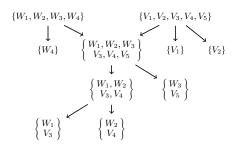


Fig. 4. The projection signature graph of Example 1.

The conditions in \mathcal{DLR}^{\pm} restrict \mathcal{DLR}^{+} in the way that multiple projections of relations may appear in a KB: intuitively, there can not be projections of a relation sharing a common attribute. Moreover, observe that in \mathcal{DLR}^{\pm} PATH \mathcal{T} is necessarily functional, due to the multitree restriction. Figure 4 shows that the projection signature graph of the knowledge base from Example 1 is indeed a multitree. Note that in the figure we have collapsed equivalent attributes in a unique equivalence class, according to the renaming schema. Furthermore, since all its projection constructs have q = 1, this knowledge base belongs to \mathcal{DLR}^{\pm} .

 \mathcal{DLR} is included in \mathcal{DLR}^{\pm} , since the projection signature graph of any \mathcal{DLR} knowledge base is always a degenerate multitree with maximum depth equal to 1. Not all the database constraints as introduced in Section 3 can be directly expressed in \mathcal{DLR}^{\pm} . While functional dependency and key axioms can be expressed directly in \mathcal{DLR}^{\pm} , equijoins, external uniqueness axioms, and identification axioms introduce projections of a relation which share common attributes, thus violating the multitree restriction. However, in \mathcal{DLR}^{\pm} it is still possible to reason over both external uniqueness and identification axioms by encoding them into a set of saturated ABoxes (as originally proposed in [Calvanese *et al.*, 2001]) and check whether there is a saturation that satisfies the constraints. Therefore, we can conclude that \mathcal{DLR}_{ifd} extended with

unary functional dependencies is included in \mathcal{DLR}^{\pm} , provided that projections of relations in the knowledge base form a multitree projection signature graph. Since (unary) functional dependencies are expressed via the inclusions of projections of relations, by constraining the projection signature graph to be a multitree, the possibility to build combinations of functional dependencies as the ones in [Calvanese *et al.*, 2001] leading to undecidability is ruled out. Concerning the ability of \mathcal{DLR}^{\pm} to capture conceptual data models, only the mapping of ORM schemas is affected by the \mathcal{DLR}^{\pm} restrictions: the projections involved in the ORM schema should satisfy the \mathcal{DLR}^{\pm} multitree restriction.

The main result of this work is that reasoning in DLR^{\pm} is an EXPTIME-complete problem. The lower bound comes by observing that DLR is a sublanguage of DLR^{\pm} , the upper bound is proved by providing a mapping from DLR^{\pm} KBs to ALCQI KBs.

5 Implementation

We have implemented the framework discussed in this paper. DLRtoOWL is a Java library fully implementing DLR^{\pm} reasoning services. The library is based on the tool ANTLR4 to parse serialised input, and on OWLAPI4 for the OWL2 encoding. The

system includes JFact, the Java version of the popular Fact++ reasoner. DLRtoOWL provides a Java \mathcal{DLR} API package to allow developers to create, manipulate, serialise, and reason with \mathcal{DLR}^{\pm} knowledge bases in their Java-based application, extending in a compatible way the standard OWL API with the \mathcal{DLR}^{\pm} TELL and ASK services. During the development of this new library we strongly focused on performance. Since the OWL encoding is only possible if we have already built the \mathcal{ALCQI} projection signature multitree, in principle the program should perform two parsing rounds: one to create the multitree and the other one to generate the OWL mapping. We faced this issue using dynamic programming: during the first (and only) parsing round we store in a data structure each axiom that we want to translate in OWL and, after building the multitree, by the dynamic programming technique we build on-the-fly a Java class which generates the required axioms.

References

- [Artale et al., 2007] A. Artale, D. Calvanese, R. Kontchakov, V. Ryzhikov, and M. Zakharyaschev. Reasoning over extended ER models. In Proc. of the 26th Int. Conf. on Conceptual Modeling (ER'07), volume 4801 of Lecture Notes in Computer Science, pages 277–292. Springer, 2007.
- [Berardi et al., 2005] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. Artificial Intelligence, 168(1–2):70–118, 2005.
- [Calvanese et al., 2001] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Identification constraints and functional dependencies in description logics. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01*, pages 155– 160. Morgan Kaufmann, 2001.
- [Calvanese et al., 2008] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive query containment and answering under description logic constraints. ACM Trans. Comput. Logic, 9(3):22:1–22:31, June 2008.
- [Calvanese *et al.*, 2014] Diego Calvanese, Wolfgang Fischl, Reinhard Pichler, Emanuel Sallinger, and Mantas Simkus. Capturing relational schemas and functional dependencies in RDFS. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1003–1011. AAAI Press, 2014.
- [Chandra and Vardi, 1985] Ashok K. Chandra and Moshe Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM Journal on Compututing*, 14(3):671–677, 1985.
- [Franconi *et al.*, 2012] Enrico Franconi, Alessandro Mosca, and Dmitry Solomakhin. ORM2: formalisation and encoding in OWL2. In *International Workshop on Fact-Oriented Modeling* (*ORM 2012*), pages 368–378, 2012.
- [Halpin and Morgan, 2008] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann, 2nd edition, 2008.
- [Kanellakis, 1990] Paris C. Kanellakis. Elements of relational database theory. In A.R. Meyer, M. Nivat, M.S. Paterson, D. Perrin, and J. van Leeuwen, editors, *The Handbook of Theoretical Computer Science*, volume B, chapter 17, pages 1075–1144. North Holland, 1990.
- [Sportelli and Franconi, 2016] Francesco Sportelli and Enrico Franconi. Formalisation of ORM derivation rules and their mapping into OWL. In *ODBASE Conference 2016*, pages 827–843, 2016.
- [Toman and Weddell, 2009] David Toman and Grant E. Weddell. Applications and extensions of PTIME description logics with functional constraints. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 948–954, 2009.