A Decidable Very Expressive Description Logic for Databases

Alessandro Artale, Enrico Franconi(✉), Rafael Peñaloza, and Francesco Sportelli

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

Abstract. We introduce $\mathcal{DLR}^+$, an extension of the n-ary propositionally closed description logic $\mathcal{DLR}$ 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. We show how a simple syntactic restriction on the appearance of projections sharing common attributes in a $\mathcal{DLR}^+$ knowledge base makes reasoning in the language decidable with the same computational complexity as $\mathcal{DLR}$. The obtained $\mathcal{DLR}^{\pm}$ n-ary description logic is able to encode more thoroughly conceptual data models such as EER, UML, and ORM.

1 Introduction

We introduce the description logic (DL) $\mathcal{DLR}^+$ extending the n-ary DL $\mathcal{DLR}$ [6], in order to capture database oriented constraints. While $\mathcal{DLR}$ is a rather expressive logic, tailored for conceptual modelling and ontology design, generalising many aspects of classical description logics and OWL, it lacks a number of expressive means relevant for database applications that can be added without increasing the complexity of reasoning—when used in a carefully controlled way. The added expressivity is motivated by the increasing use of description logics as an abstract conceptual layer (an *ontology*) over relational databases. For example, the $\mathcal{DLR}$ family of description logics is used to formalise and perform reasoning in the ORM conceptual modelling language for database design (adopted by Microsoft in Visual Studio) [8,15].

We remind that a $\mathcal{DLR}$ knowledge base, as defined in [6], can express axioms with (i) propositional combinations of concepts and (compatible) n-ary relations – as opposed to just binary roles as in classical description logics and OWL, (ii) concepts as unary projections of n-ary relations – generalising the existential operator over binary roles in classical description logics and OWL, and (iii) relations with a selected typed component.

C. d'Amato et al. (Eds.): ISWC 2017, Part I, LNCS 10587, pp. 37–52, 2017.
DOI: 10.1007/978-3-319-68288-4_3

As an example of $\mathcal{DLR}$, in a knowledge base where `Pilot` and `RacingCar` are concepts and `DrivesCar`, `DrivesMotorbike`, `DrivesVehicle` are binary relations, the following statements:

$$\texttt{Pilot} \sqsubseteq \exists[1]\sigma_{2:\texttt{RacingCar}}\texttt{DrivesCar}$$
$$\texttt{DrivesCar} \sqcup \texttt{DrivesMotorbike} \sqsubseteq \texttt{DrivesVehicle}$$

assert that a pilot drives a racing car and that driving a car or a motorbike implies driving a vehicle.

The language we propose here, $\mathcal{DLR}^+$, extends $\mathcal{DLR}$ in the following ways.

- 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* of objects, i.e., tuples where each component is identified by an attribute and not by its position in the tuple (see, e.g., [11]). For example, the relation `Employee` may have the signature:

  $$\texttt{Employee}(\texttt{firstname}, \texttt{lastname}, \texttt{dept}, \texttt{deptAddr}),$$

  and an instance of `Employee` could be the tuple:

  $$\langle \texttt{firstname} : \texttt{John}, \texttt{lastname} : \texttt{Doe}, \texttt{dept} : \texttt{Purchase}, \texttt{deptAddr} : \texttt{London} \rangle.$$

- Attributes can be *renamed*, for example to recover the positional attributes:

  $$\texttt{firstname}, \texttt{lastname}, \texttt{dept}, \texttt{deptAddr} \rightleftarrows 1, 2, 3, 4.$$

- *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::

  $$\pi[\texttt{firstname}, \texttt{lastname}]\texttt{Employee} \sqsubseteq \texttt{Person},$$
  $$\texttt{firstname}, \texttt{lastname} \rightleftarrows \texttt{name}, \texttt{surname}.$$

- The *objectification* of a relation (also known as *reification*) is a concept whose instances are unique *object 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 1-to-1 correspondence between pairs of values of the attributes `firstname`, `lastname` and `EmployeeC` instances:

  $$\texttt{EmployeeC} \equiv \textcircled{\scriptsize\bigcirc} \exists[\texttt{firstname}, \texttt{lastname}]\texttt{Employee}.$$

  Consider the relations with the following signatures:

  $$\texttt{DrivesCar}(\texttt{name}, \texttt{surname}, \texttt{car}), \quad \texttt{OwnsCar}(\texttt{name}, \texttt{surname}, \texttt{car}),$$

and assume that anybody driving a car also owns it: $\texttt{DrivesCar} \sqsubseteq \texttt{OwnsCar}$. The *locally* objectified events of driving and owning, defined as

$$\texttt{CarDrivingEvent} \equiv \textcircled{\cdot}\,\texttt{DrivesCar}, \quad \texttt{CarOwningEvent} \equiv \textcircled{\cdot}\,\texttt{OwnsCar},$$

do not imply that a car driving event by a person is the owning event by the same person and the same car: $\texttt{CarDrivingEvent} \not\sqsubseteq \texttt{CarOwningEvent}$. Indeed, they are even disjoint: $\texttt{CarDrivingEvent} \sqcap \texttt{CarOwningEvent} \sqsubseteq \bot$.

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

$$\pi[\texttt{firstname}, \texttt{lastname}]\texttt{Employee} \sqsubseteq \pi^{\leqslant 1}[\texttt{firstname}, \texttt{lastname}]\texttt{Employee},$$

and that the attribute $\texttt{deptAddr}$ functionally depends on the attribute $\texttt{dept}$ within the relation $\texttt{Employee}$:

$$\exists[\texttt{dept}]\texttt{Employee} \sqsubseteq \exists^{\leqslant 1}[\texttt{dept}]\,(\pi[\texttt{dept}, \texttt{deptAddr}]\texttt{Employee})\,.$$

While $\mathcal{DLR}^+$ turns out to be undecidable, we show how a simple syntactic condition on the appearance of projections sharing common attributes in a 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 Sect. 6 the implementation of an API for the reasoning services in $\mathcal{DLR}^{\pm}$.

2 The Description Logic $\mathcal{DLR}^+$

We start by introducing the syntax of $\mathcal{DLR}^+$. 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 $n \geqslant 2$.

$$\begin{array}{lcl}
C & \rightarrow & CN \mid \neg C \mid C_1 \sqcap C_2 \mid \exists^{\geqslant q}[U_i]R \mid \circledcirc R \mid \textcircled{\cdot}\,RN \\
R & \rightarrow & RN \mid R_1 \backslash R_2 \mid R_1 \sqcap R_2 \mid R_1 \sqcup R_2 \mid \sigma_{U_i:C} R \mid \pi^{\leqslant q}[U_1, \ldots, U_k]R \\
\varphi & \rightarrow & C_1 \sqsubseteq C_2 \mid R_1 \sqsubseteq R_2 \mid CN(o) \mid RN(U_1{:}o_1, \ldots, U_n{:}o_n) \mid o_1 = o_2 \mid o_1 \neq o_2 \\
\vartheta & \rightarrow & U_1 \rightleftarrows U_2
\end{array}$$

Fig. 1. The syntax of $\mathcal{DLR}^+$.

The syntax of concepts C, relations R, formulas φ, and attribute renaming axioms ϑ is given in Fig. 1, where $CN \in \mathcal{C}$, $RN \in \mathcal{R}$, $U \in \mathcal{U}$, $o \in \mathcal{O}$, q is a positive integer and $2 \leqslant k < \text{ARITY}(R)$. The *arity* of a relation R is the number of the attributes in its signature; i.e., $\text{ARITY}(R) = |\tau(R)|$, with the relation signature function τ extended to complex relations as in Fig. 2. Note that it is possible that the same attribute appears in the signature of different relations.

$$
\begin{array}{rll}
\tau(R_1 \backslash R_2) = & \tau(R_1) & \\
\tau(R_1 \sqcap R_2) = & \tau(R_1) & \text{if } \tau(R_1) = \tau(R_2) \\
\tau(R_1 \sqcup R_2) = & \tau(R_1) & \text{if } \tau(R_1) = \tau(R_2) \\
\tau(\sigma_{U_i:C} R) = & \tau(R) & \text{if } U_i \in \tau(R) \\
\tau(\pi^{\leqslant q}_{\geqslant}[U_1, \ldots, U_k]R) = & \{U_1, \ldots, U_k\} & \text{if } \{U_1, \ldots, U_k\} \subset \tau(R) \\
\text{undefined} & & \text{otherwise}
\end{array}
$$

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

As mentioned in the introduction, the $\mathcal{DLR}^+$ constructors added to $\mathcal{DLR}$ are the *local* and *global objectification* ($\bigodot RN$ and $\circledcirc R$, respectively); *relation projections* with the possibility to count the projected tuples ($\pi^{\leqslant q}_{\geqslant}[U_1, \ldots, U_k]R$), and *renaming axioms* over attributes ($U_1 \rightleftarrows U_2$). Note that local objectification ($\bigodot R$) can be applied to relation names, while global objectification ($\circledcirc RN$) can be applied to arbitrary relation expressions. We use the standard abbreviations:

$$
\bot = C \sqcap \neg C, \quad \top = \neg\bot, \quad C_1 \sqcup C_2 = \neg(\neg C_1 \sqcap \neg C_2), \quad \exists[U_i]R = \exists^{\geqslant 1}[U_i]R,
$$
$$
\exists^{\leqslant q}[U_i]R = \neg(\exists^{\geqslant q+1}[U_i]R), \quad \pi[U_1, \ldots, U_k]R = \pi^{\geqslant 1}[U_1, \ldots, U_k]R.
$$

A $\mathcal{DLR}^+$ *TBox* $\mathcal{T}$ is a finite set of *concept inclusion* axioms of the form $C_1 \sqsubseteq C_2$ and *relation inclusion* axioms of the form $R_1 \sqsubseteq R_2$. We use $X_1 \equiv X_2$ as a shortcut for the two axioms $X_1 \sqsubseteq X_2$ and $X_2 \sqsubseteq X_1$. 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 unrestricted TBox axioms. A $\mathcal{DLR}^+$ *renaming schema* $\Re$ is a finite set of renaming axioms of the form $U_1 \rightleftarrows U_2$. We use the shortcut $U_1 \ldots U_n \rightleftarrows U'_1 \ldots U'_n$ to group many renaming axioms with the meaning that $U_i \rightleftarrows U'_i$ for all $i = 1, \ldots, n$. 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 $\Re$.

The renaming operator $\rightleftarrows$ is an equivalence relation over the attributes $\mathcal{U}$, ($\rightleftarrows$,$\mathcal{U}$). The partitioning of $\mathcal{U}$ into equivalence classes induced by a renaming schema is meant to represent the alternative ways to name attributes in the knowledge base. A unique *canonical representative* for each equivalence class is chosen to replace all the attributes in the class throughout the knowledge base. From now on we assume that a knowledge base is consistently rewritten by

substituting each attribute with its canonical representative. After this rewriting, the renaming schema does not play any role in the knowledge base. We allow only *arity-preserving* renaming schemas, i.e., there is no equivalence class containing two attributes from the same relation signature.

As shown in the introduction, the renaming schema is useful to reconcile the named attribute perspective and the positional perspective on relations. It is also important to enforce *union compatibility* among relations involved in relation inclusion axioms, and among relations involved in $\sqcap$- and $\sqcup$-set expressions. Two relations are *union compatible* (w.r.t. a renaming schema) if they have the same signature (up to the attribute renaming induced by the renaming schema). Indeed, as it will be clear from the semantics, a relation inclusion axiom involving non union compatible relations would always be false, and a $\sqcap$- and $\sqcup$-set expression involving non union compatible relations would always be empty.

The semantics of $\mathcal{DLR}^+$ uses the notion of *labelled tuples* over a countable potentially infinite domain Δ. Given a set of labels $\mathcal{X} \subseteq \mathcal{U}$ an *$\mathcal{X}$-labelled tuple over Δ* (or *tuple* for short) is a *total* function $t\colon \mathcal{X} \to \Delta$. For $U \in \mathcal{X}$, we write $t[U]$ to refer to the domain element $d \in \Delta$ labelled by U. Given $d_1, \ldots, d_n \in \Delta$, the expression $\langle U_1\colon d_1, \ldots, U_n\colon d_n\rangle$ stands for the tuple t defined on the set of labels $\{U_1, \ldots, U_n\}$ such that $t[U_i] = d_i$, for $1 \leqslant 1 \leqslant n$. The *projection* of the tuple t over the attributes $U_1, \ldots, U_k$ is the function t restricted to be undefined for the labels not in $U_1, \ldots, U_k$, and it is denoted by $t[U_1, \ldots, U_k]$. The relation signature function τ is extended to labelled tuples to obtain the set of labels on which a tuple is defined. $T_\Delta(\mathcal{X})$ denotes the set of all $\mathcal{X}$-labelled tuples over Δ, for $\mathcal{X} \subseteq \mathcal{U}$, and we overload this notation by denoting with $T_\Delta(\mathcal{U})$ the set of all possible tuples with labels within the whole set of attributes $\mathcal{U}$.

A $\mathcal{DLR}^+$ *interpretation* is a tuple $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}}, \imath, L)$ consisting of a nonempty countable potentially infinite *domain* Δ specific to $\mathcal{I}$, an *interpretation function* $\cdot^{\mathcal{I}}$, a *global objectification function* $\imath$, and a family L containing one *local objectification function* ℓ_{RN_i} for each named relation $RN_i \in \mathcal{R}$. The global objectification

$$
\begin{aligned}
(\neg C)^{\mathcal{I}} &= \top^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists^{\geqslant q}[U_i]R)^{\mathcal{I}} &= \{d \in \Delta \mid \ \big|\{t \in R^{\mathcal{I}} \mid t[U_i] = d\}\big| \geqslant q\} \\
(\circledcirc\, R)^{\mathcal{I}} &= \{d \in \Delta \mid d = \imath(t) \wedge t \in R^{\mathcal{I}}\} \\
(\odot\, RN)^{\mathcal{I}} &= \{d \in \Delta \mid d = \ell_{RN}(t) \wedge t \in RN^{\mathcal{I}}\} \\
(R_1 \backslash R_2)^{\mathcal{I}} &= R_1^{\mathcal{I}} \backslash R_2^{\mathcal{I}} \\
(R_1 \sqcap R_2)^{\mathcal{I}} &= R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}} \\
(R_1 \sqcup R_2)^{\mathcal{I}} &= \{t \in R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}} \mid \tau(R_1) = \tau(R_2)\} \\
(\sigma_{U_i:C} R)^{\mathcal{I}} &= \{t \in R^{\mathcal{I}} \mid t[U_i] \in C^{\mathcal{I}}\} \\
(\pi^{\lesseqgtr q}[U_1, \ldots, U_k]R)^{\mathcal{I}} &= \{\langle U_1 : d_1, \ldots, U_k : d_k\rangle \in T_\Delta(\{U_1, \ldots, U_k\}) \mid \\
&\qquad 1 \leqslant \big|\{t \in R^{\mathcal{I}} \mid t[U_1] = d_1, \ldots, t[U_k] = d_k\}\big| \lesseqgtr q\}
\end{aligned}
$$

Fig. 3. The semantics of $\mathcal{DLR}^+$ expressions.

function is an injective function, $\imath : T_\Delta(\mathcal{U}) \rightarrow \Delta$, associating a *unique* global identifier to each tuple. The local objectification functions, $\ell_{RN_i} : T_\Delta(\mathcal{U}) \rightarrow \Delta$, 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 $\tau(RN)$-labelled tuples over Δ to each relation name RN, $RN^{\mathcal{I}} \subseteq T_\Delta(\tau(RN))$. 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 Fig. 3. Notice that the construct $\pi^{\overset{\leq}{>}q}[U_1, \ldots, U_k]R$ is interpreted as a classical projection over a relation, thus including only tuples belonging to the relation.

The interpretation $\mathcal{I}$ satisfies the concept inclusion axiom $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, and the relation inclusion axiom $R_1 \sqsubseteq R_2$ 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 $\langle U_1 : o_1^{\mathcal{I}}, \ldots, U_n : o_n^{\mathcal{I}} \rangle \in RN^{\mathcal{I}}$, 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. $\mathcal{I}$ is a *model* of the knowledge base $(\mathcal{T}, \mathcal{A}, \Re)$ if it satisfies all the axioms in the TBox $\mathcal{T}$ and in the ABox $\mathcal{A}$, once the knowledge base has been rewritten according to the renaming schema.

Example 1. Consider the relation names R_1, R_2 with $\tau(R_1) = \{W_1, W_2, W_3, W_4\}$, $\tau(R_2) = \{V_1, V_2, V_3, V_4, V_5\}$, and a knowledge base with the renaming axiom $W_1W_2W_3 \rightleftarrows V_3V_4V_5$ and a TBox $\mathcal{T}_{\mathsf{exa}}$:

$$\pi[W_1, W_2]R_1 \sqsubseteq \pi^{\leqslant 1}[W_1, W_2]R_1 \tag{1}$$

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

$$\pi[W_1, W_2, W_3]R_1 \sqsubseteq \pi[V_3, V_4, V_5]R_2. \tag{3}$$

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. □

KB satisfiability refers to the problem of deciding the existence of a model of a given knowledge base; *concept satisfiability* (resp. *relation satisfiability*) is the problem of deciding whether there is a model of the knowledge base with a non-empty interpretation of a given concept (resp. relation). A knowledge base *entails* (or *logically implies*) an axiom if all models of the knowledge base are also models of the axiom. For instance, it is easy to see that the TBox in Example 1 entails that V_3, V_4 are a key for R_2:

$$\mathcal{T}_{\mathsf{exa}} \models \pi[V_3, V_4]R_2 \sqsubseteq \pi^{\leqslant 1}[V_3, V_4]R_2,$$

and that axiom (2) is redundant in $\mathcal{T}_{\mathsf{exa}}$. The decision problems in $\mathcal{DLR}^+$ can be all reduced to KB satisfiability.

Lemma 2. *In $\mathcal{DLR}^+$, concept and relation satisfiability and entailment are reducible to KB satisfiability.*

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* (namely inclusion axioms among arbitrary projections of relations), *equijoins*, *functional dependency* axioms, *key* and *foreign 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 signatures $\tau(R_1) = \{U, U_1, \ldots, U_{n_1}\}$ and $\tau(R_2) = \{V, V_1, \ldots, V_{n_2}\}$; their equijoin over U and V is the relation $R = R_1 \underset{U=V}{\bowtie} R_2$ with signature $\tau(R) = \tau(R_1) \cup \tau(R_2) \setminus \{V\}$, which is expressed by the $\mathcal{DLR}^+$ axioms:

$$\pi[U, U_1, \ldots, U_{n_1}]R \equiv \sigma_{U:(\exists[U]R_1 \sqcap \exists[V]R_2)}R_1$$
$$\pi[V, V_1, \ldots, V_{n_2}]R \equiv \sigma_{V:(\exists[U]R_1 \sqcap \exists[V]R_2)}R_2$$
$$U \rightleftarrows V.$$

A *functional dependency* axiom ($R : U_1 \ldots U_j \rightarrow U$) (also called *internal uniqueness* axiom [9]) states that the values of the attributes $U_1 \ldots U_j$ 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 $s, t \in R^{\mathcal{I}}$, $s[U_1] = t[U_1], \ldots, s[U_j] = t[U_j]$ imply $s[U] = t[U]$. Functional dependencies can be expressed in $\mathcal{DLR}^+$, assuming that $\{U_1, \ldots, U_j, U\} \subseteq \tau(R)$, with the axiom:

$$\pi[U_1, \ldots, U_j]R \sqsubseteq \pi^{\leqslant 1}[U_1, \ldots, U_j](\pi[U_1, \ldots, U_j, U]R).$$

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

$$\pi[U_1, \ldots, U_j]R \sqsubseteq \pi^{\leqslant 1}[U_1, \ldots, U_j]R.$$

A *foreign key* is the obvious result of an inclusion dependency together with a key constraint involving the foreign key attributes.

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 [9]. This can be expressed in $\mathcal{DLR}^+$ with the axioms:

$$R \equiv R_1 \underset{U^1=U^2}{\bowtie} \cdots \underset{U^{h-1}=U^h}{\bowtie} R_h$$
$$R : U^1_1, \ldots, U^1_{n_1}, \ldots, U^h_1, \ldots, U^h_{n_h} \rightarrow U^1$$

where $\tau(R_i) = \{U^i, U^i_1, \ldots, U^i_{n_i}\}, 1 \leq i \leq h$, and R is a new relation name with $\tau(R) = \{U^1, U^1_1, \ldots, U^1_{n_1}, \ldots, U^h_1, \ldots, U^h_{n_h}\}$.

Identification axioms as defined in $\mathcal{DLR}_{ifd}$ [4] (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 description logics family $\mathcal{CFD}$ [16]—can be expressed in $\mathcal{DLR}^+$ as identification axioms involving joined sequences of functional binary relations. $\mathcal{DLR}^+$ also captures the *tree-based identification constraints (tid)* introduced in [5] to express functional dependencies 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 database models, and to provide the foundations for their reasoning tasks. The EER mapping as introduced in [1] can be extended to deal with multi-attribute keys (by using identification axioms) and named roles in relations; the ORM mapping as introduced in [8,15] 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 [3] can be fixed to deal properly with association classes (by using local objectification) and named roles in associations.

Aside from conceptual modelling, $\mathcal{DLR}^+$ could be studied in relation to other tasks relevant for database scenarios, such as query answering [6], constraint checking with respect to a partially closed world (i.e., with DBoxes [13]), inconsistent database repairing, etc. In this paper, we focus just on the basic consistency and entailment reasoning tasks.

4 The $\mathcal{DLR}^{\pm}$ Fragment of $\mathcal{DLR}^+$

Since a $\mathcal{DLR}^+$ knowledge base can express inclusions and functional dependencies, the entailment problem is undecidable [7]. Thus, in this section we present $\mathcal{DLR}^{\pm}$, a decidable syntactic fragment of $\mathcal{DLR}^+$ limiting the coexistence of relation projections in a knowledge base.

Given a $\mathcal{DLR}^+$ knowledge base $\mathcal{KB} = (\mathcal{T}, \mathcal{A}, \Re)$, we define the *projection signature of* $\mathcal{KB}$ as the set $\mathcal{S}$ containing the signatures $\tau(RN)$ of all 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, $\mathcal{S}$ is the smallest set such that (i) $\tau(RN) \in \mathcal{S}$ for all $RN \in \mathcal{R}$; (ii) $\{U\} \in \mathcal{S}$ for all $U \in \mathcal{U}$; and (iii) $\{U_1, \ldots, U_k\} \in \mathcal{S}$ for all $\pi^{\leq q}[V_1, \ldots, V_k]R$ appearing as sub-formulas in $\mathcal{T}$ and $V_i \in [U_i]_{\Re}$ for $1 \leqslant i \leqslant k$.

The *projection signature graph* of $\mathcal{KB}$ is the directed acyclic graph corresponding to the Hasse diagram of $\mathscr{T}$ ordered by the proper subset relation $\supset$, whose sinks are the attribute singletons $\{U\}$. We call this graph $(\supset, \mathscr{T})$. Given a set of attributes $\tau = \{U_1, \ldots, U_k\} \subseteq \mathcal{U}$, the *projection signature graph dominated by* τ, denoted as $\mathscr{T}_\tau$, is the sub-graph of $(\supset, \mathscr{T})$ with τ as root and containing all the nodes reachable from τ. Given two sets of attributes $\tau_1, \tau_2 \subseteq \mathcal{U}$, $\text{PATH}_{\mathscr{T}}(\tau_1, \tau_2)$ denotes the set of paths in $(\supset, \mathscr{T})$ between τ_1 and τ_2. Note that, $\text{PATH}_{\mathscr{T}}(\tau_1, \tau_2) = \varnothing$ both when a path does not exist and when $\tau_1 \subseteq \tau_2$. The notation $\text{CHILD}_{\mathscr{T}}(\tau_1, \tau_2)$ means that τ_2 is a child (i.e., a direct descendant) of τ_1 in $(\supset, \mathscr{T})$. We now introduce $\mathcal{DLR}^{\pm}$ as follows.

Definition 3. *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* $\mathscr{T}_\tau$ *is a tree; and*
2. *for every projection construct* $\pi^{\leq q}[U_1, \ldots, U_k]R$ *and every concept expression of the form* $\exists^{\geqslant q}[U_1]R$ *appearing in* $\mathcal{T}$*, if* $q > 1$ *then the length of the path* $\text{PATH}_{\mathscr{T}}(\tau(R), \{U_1, \ldots, U_k\})$ *is 1.*

The first condition in $\mathcal{DLR}^{\pm}$ restrict $\mathcal{DLR}^{+}$ in the way that multiple projections of relations may appear in a knowledge base: intuitively, there cannot be different projections sharing a common attribute. Moreover, observe that in $\mathcal{DLR}^{\pm}$ $\text{PATH}_{\mathscr{T}}$ is necessarily functional, due to the multitree restriction. By relaxing the first condition the language becomes undecidable, as we mentioned at the beginning of this Section. The second condition is also necessary to prove decidability of $\mathcal{DLR}^{\pm}$; however, we do not know whether this condition could be relaxed while preserving decidability.

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

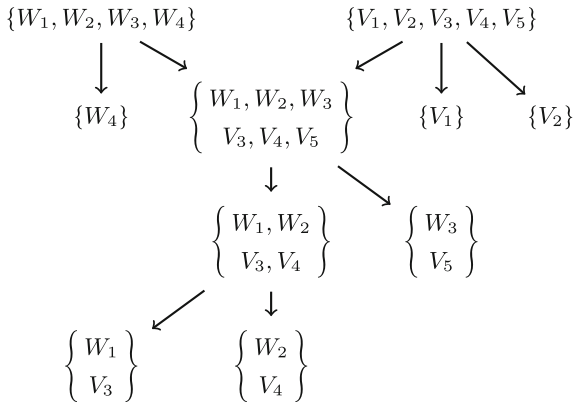

Fig. 4. The projection signature graph of Example 1.

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 Sect. 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. For example, the axioms for capturing an equijoin between two relations, R_1, R_2 would generate a projection signature graph with the signatures of R_1, R_2 as projections of the signature of the join relation R sharing the attribute on which the join is performed, thus violating condition 1.

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 [4]) 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 [4] leading to undecidability is ruled out.

Note that the *non-conflicting keys* sufficient condition guaranteeing the decidability of inclusion dependencies and keys of [12] is in conflict with our more restrictive requirement: indeed [12] allow for overlapping projections, but the considered datalog language is not comparable to $\mathcal{DLR}^{+}$. In general, description logic based languages, such as $\mathcal{DLR}^{+}$, and datalog based languages, such as the language proposed in [12], are incomparable in terms of expressiveness, due to the inability of description logics to distinguish non-tree models in the TBox. Note that, unlike the typical restrictions of datalog-like languages, there is no problem in stating arbitrary cyclic dependencies in relation inclusion axioms involving projections on the left and right hand sides.

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: $\mathcal{DLR}^{\pm}$ is able to correctly express an ORM schema if the projections involved in the schema satisfy the $\mathcal{DLR}^{\pm}$ multitree restriction.

5 Mapping $\mathcal{DLR}^{\pm}$ to $\mathcal{ALCQI}$

This section shows constructively the main technical result of this paper, i.e., that reasoning in $\mathcal{DLR}^{\pm}$ is an ExpTime-complete problem. The lower bound is clear by observing that $\mathcal{DLR}$ is a sublanguage of $\mathcal{DLR}^{\pm}$. More challenging is the upper bound obtained by providing a mapping from $\mathcal{DLR}^{\pm}$ knowledge bases

to $\mathcal{ALCQI}$ knowledge bases—a propositionally complete description logic with qualified number restrictions $\exists^{\geqslant q} R.\, C$, and inverse roles R^- (see [2] for more details). We adapt and extend the mapping presented for $\mathcal{DLR}$ in [6], with the modifications proposed by [10] to deal with ABoxes without the unique name assumption.

We recall that the renaming schema $\Re$ does not play any role since we assumed that a $\mathcal{DLR}^{\pm}$ knowledge base is rewritten by choosing a single canonical representative for each equivalence class of attributes induced by $\Re$. Thus, we consider $\mathcal{DLR}^{\pm}$ knowledge bases as pairs of TBox and ABox axioms.

$$
\begin{aligned}
(\neg C)^{\dagger} &= \neg C^{\dagger} \\
(C_1 \sqcap C_2)^{\dagger} &= C_1^{\dagger} \sqcap C_2^{\dagger} \\
(\exists^{\geqslant q}[U_i]R)^{\dagger} &= \begin{cases} \exists^{\geqslant q} \left(\text{PATH}_{\mathscr{T}}(\tau(R), \{U_i\})^{\dagger}\right)^{-}.\, R^{\dagger}, \text{ if } \text{PATH}_{\mathscr{T}}(\tau(R), \{U_i\}) \neq \varnothing \\ \bot, \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \\
(\circledcirc\, R)^{\dagger} &= R^{\dagger} \\
(\odot\, RN)^{\dagger} &= A^{l}_{RN} \\
\\
(R_1 \backslash R_2)^{\dagger} &= R_1^{\dagger} \sqcap \neg R_2^{\dagger} \\
(R_1 \sqcap R_2)^{\dagger} &= R_1^{\dagger} \sqcap R_2^{\dagger} \\
(R_1 \sqcup R_2)^{\dagger} &= \begin{cases} R_1^{\dagger} \sqcup R_2^{\dagger}, \text{ if } \tau(R_1) = \tau(R_2) \\ \bot, \qquad \text{otherwise} \end{cases} \\
(\sigma_{U_i:C} R)^{\dagger} &= \begin{cases} R^{\dagger} \sqcap \forall \text{PATH}_{\mathscr{T}}(\tau(R), \{U_i\})^{\dagger}.\, C^{\dagger}, \text{ if } \text{PATH}_{\mathscr{T}}(\tau(R), \{U_i\}) \neq \varnothing \\ \bot, \qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \\
(\pi^{\leqslant q}[U_1, \ldots, U_k]R)^{\dagger} &= \begin{cases} \exists^{\geqslant 1, \leqslant q} \left(\text{PATH}_{\mathscr{T}}(\tau(R), \{U_1, \ldots, U_k\})^{\dagger}\right)^{-}.\, R^{\dagger}, \\ \qquad\qquad\qquad\qquad \text{if } \text{PATH}_{\mathscr{T}}(\tau(R), \{U_1, \ldots, U_k\}) \neq \varnothing \\ \bot, \qquad\qquad\qquad \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 5. The mapping to $\mathcal{ALCQI}$ for concept and relation expressions.

We first introduce a mapping function $\cdot^{\dagger}$ from $\mathcal{DLR}^{\pm}$ concepts and relations to $\mathcal{ALCQI}$ concepts. The function $\cdot^{\dagger}$ maps each concept name CN and each relation name RN appearing in the $\mathcal{DLR}^{\pm}$ KB to the $\mathcal{ALCQI}$ concept names CN and A_{RN}, respectively. The latter can be informally understood as the "global" reification of RN. For each relation name RN, the $\mathcal{ALCQI}$ signature also includes a concept name A^{l}_{RN} and a role name Q_{RN} to capture local objectification. The mapping $\cdot^{\dagger}$ is extended to concept and relation expressions as illustrated in Fig. 5, where the notation $\exists^{\geqslant 1, \leqslant q} R.\, C$ is a shortcut for the conjunction $\exists R.\, C \sqcap \exists^{\leqslant q} R.\, C$.

The mapping crucially uses the projection signature graph to map projections and selections, by accessing paths in the projection signature graph $(\supset, \mathscr{T})$ associated to the $\mathcal{DLR}^{\pm}$ KB. If there is a path $\text{PATH}_{\mathscr{T}}(\tau, \tau') = \tau, \tau_1, \ldots, \tau_n, \tau'$

from τ to τ' in $\mathscr{T}$, then the $\mathcal{ALCQI}$ signature contains role names $Q_{\tau'}, Q_{\tau_i}$, for $i = 1, \ldots, n$, and the following role chain expression is generated by the mapping:

$$\text{PATH}_{\mathscr{T}}(\tau, \tau')^{\dagger} = Q_{\tau_1} \circ \ldots \circ Q_{\tau_n} \circ Q_{\tau'},$$

In particular, the mapping uses the following notation: the inverse role chain $(R_1 \circ \ldots \circ R_n)^-$, for R_i a role name, stands for the chain $R_n^- \circ \ldots \circ R_1^-$, with R_i^- an inverse role, the expression $\exists^{\leqslant 1} R_1 \circ \ldots \circ R_n . C$ stands for the $\mathcal{ALCQI}$ concept expression $\exists^{\leqslant 1} R_1 . \ldots . \exists^{\leqslant 1} R_n . C$ and $\forall R_1 \circ \ldots \circ R_n . C$ for the $\mathcal{ALCQI}$ concept expression $\forall R_1 . \ldots . \forall R_n . C$. Thus, since $\mathcal{DLR}^{\pm}$ restricts to $q = 1$ the cardinalities on any path of length strictly greater than 1 (see condition 2 in Definition 3), the above notation shows that we remain within the $\mathcal{ALCQI}$ syntax when the mapping applies to cardinalities. If, e.g., we need to map the $\mathcal{DLR}^{\pm}$ cardinality constraint $\exists^{\leqslant q}[U_i]R$ with $q > 1$, then, to stay within the $\mathcal{ALCQI}$ syntax, U_i must not be mentioned in any other projection in such a way that $|\text{PATH}_{\mathscr{T}}(\tau(R), \{U_i\})| = 1$. Finally, notice that the mapping introduces a concept name $A_{RN}^{\tau_i}$ for each projected signature τ_i in the projection signature graph dominated by $\tau(RN)$, i.e., $\tau_i \in \mathscr{T}_{\tau(RN)}$, informally to capture the global reifications of the various projections of RN in the given KB. We also use the shortcut A_{RN} which stands for $A_{RN}^{\tau(RN)}$.

Intuitively, each node in the projection signature graph associated to a $\mathcal{DLR}^{\pm}$ KB denotes a relation projection and the mapping reifies each of these projections. The target $\mathcal{ALCQI}$ signature resulting from mapping the $\mathcal{DLR}^{\pm}$ KB of Example 1 is partially presented in Fig. 6, together with the projection signature graph (showed in Fig. 4). Each node of the graph is labelled with the corresponding global reification concept ($A_{R_i}^{\tau_j}$), for each $R_i \in \mathcal{R}$ and each projected signature

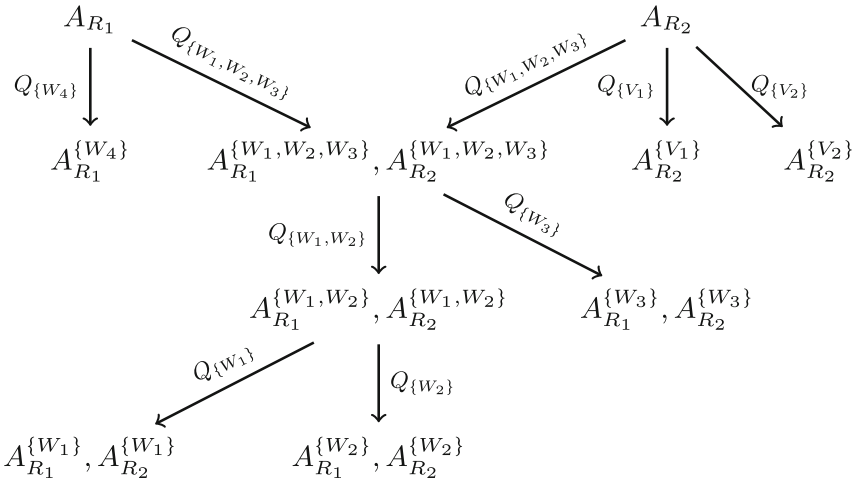

Fig. 6. The $\mathcal{ALCQI}$ signature generated by $\mathcal{T}_{\text{exa}}$.

τ_j in the projection signature graph dominated by $\tau(R_i)$, while the edges are labelled by the roles (Q_{τ_i}) needed for the reification.

To better clarify the need for the path function in the mapping, notice that each $\mathcal{DLR}^{\pm}$ relation is reified according to the decomposition dictated by the projection signature graph it dominates. Thus, to access, e.g., an attribute U_j of a $\mathcal{DLR}^{\pm}$ relation R_i it is necessary to follow the path through the projections that use the attribute. Such a path, from the node denoting the whole signature of the relation, $\tau(R_i)$, to the node denoting the attribute U_j is returned by the $\text{PATH}_{\mathscr{T}}(\tau(R_i), U_j)$ function. For example, considering the example in Fig. 6, to access the attribute W_1 of the relation R_2 in the expression $(\sigma_{W_1:C} R_2)$, the mapping of the path $\text{PATH}_{\mathscr{T}}(\tau(R_2), \{W_1\})^{\dagger}$ is equal to the role chain $Q_{\{W_1,W_2,W_3\}} \circ Q_{\{W_1,W_2\}} \circ Q_{\{W_1\}}$, so that $(\sigma_{W_1:C} R_2)^{\dagger} = A_{R_2} \sqcap \forall Q_{\{W_1,W_2,W_3\}}.\forall Q_{\{W_1,W_2\}}.\forall Q_{\{W_1\}}.C$. Similar considerations can be done when mapping cardinalities over relation projections.

Figures 7 and 8 present in details the mapping of a $\mathcal{DLR}^{\pm}$ KB into a KB in $\mathcal{ALCQI}$. Let $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{DLR}^{\pm}$ KB with signature $(\mathcal{C}, \mathcal{R}, \mathcal{O}, \mathcal{U}, \tau)$. The mapping $\gamma(\mathcal{KB})$ is assumed to be unsatisfiable (i.e., it contains the axiom $\top \sqsubseteq \bot$) if the ABox contains the relation assertion $RN(t)$ with $\tau(RN) \neq \tau(t)$, for some relation $RN \in \mathcal{R}$ and some tuple t. Otherwise, $\gamma(\mathcal{KB}) = (\gamma(\mathcal{T}), \gamma(\mathcal{A}))$ defines the mapped $\mathcal{ALCQI}$ KB.

Intuitively, γ_{dsj} ensures that relations with different signatures are disjoint, thus, e.g., enforcing the union compatibility. The axioms in γ_{rel} introduce classical reification axioms for each relation and its relevant projections. The axioms in γ_{lobj} make sure that each local objectification differs from the global one while each role Q_{RN} defines a bijection.

To translate the ABox, we first map each individual $o \in \mathcal{O}$ in the $\mathcal{DLR}^{\pm}$ ABox $\mathcal{A}$ to an $\mathcal{ALCQI}$ individual o. Each tuple in relation instance axioms occurring in $\mathcal{A}$ is mapped via an injective function ξ to a distinct individual. That is,

$$
\begin{aligned}
\gamma(\mathcal{T}) = \gamma_{dsj} \;\cup\; & \bigcup_{RN \in \mathcal{R}} \gamma_{rel}(RN) \;\cup\; \bigcup_{RN \in \mathcal{R}} \gamma_{lobj}(RN) \;\cup \\
& \bigcup_{C_1 \sqsubseteq C_2 \in \mathcal{KB}} C_1^{\dagger} \sqsubseteq C_2^{\dagger} \;\cup\; \bigcup_{R_1 \sqsubseteq R_2 \in \mathcal{KB}} R_1^{\dagger} \sqsubseteq R_2^{\dagger} \\
\gamma_{dsj} = \; & \big\{ A_{RN_1}^{\tau_i} \sqsubseteq \neg A_{RN_2}^{\tau_j} \mid RN_1, RN_2 \in \mathcal{R}, \\
& \qquad \tau_i \in \mathscr{T}_{\tau(RN_1)}, \tau_j \in \mathscr{T}_{\tau(RN_2)}, |\tau_i| \geqslant 2, |\tau_j| \geqslant 2, \tau_i \neq \tau_j \big\} \\
\gamma_{rel}(RN) = & \bigcup_{\tau_i \in \mathscr{T}_{\tau(RN)}} \bigcup_{\text{CHILD}_{\mathscr{T}}(\tau_i, \tau_j)} \big\{ A_{RN}^{\tau_i} \sqsubseteq \exists Q_{\tau_j}.A_{RN}^{\tau_j},\; \exists^{\geqslant 2} Q_{\tau_j}.\top \sqsubseteq \bot \big\} \\
\gamma_{lobj}(RN) = \; & \{ A_{RN} \sqsubseteq \exists Q_{RN}.A_{RN}^{l},\; \exists^{\geqslant 2} Q_{RN}.\top \sqsubseteq \bot, \\
& \; A_{RN}^{l} \sqsubseteq \exists Q_{RN}^{-}.A_{RN},\; \exists^{\geqslant 2} Q_{RN}^{-}.\top \sqsubseteq \bot \}.
\end{aligned}
$$

Fig. 7. The mapping into a $\mathcal{ALCQI}$ KB.

$$\gamma(\mathcal{A}) = \{CN^{\dagger}(o) \mid CN(o) \in \mathcal{A}\} \cup \tag{4}$$
$$\{o_1 \neq o_2 \mid o_1 \neq o_2 \in \mathcal{A}\} \cup \{o_1 = o_2 \mid o_1 = o_2 \in \mathcal{A}\} \cup \tag{5}$$
$$\{A^{\tau_i}_{RN}(\xi(t[\tau_i])) \mid RN(t) \in \mathcal{A} \text{ and } \tau_i \in \mathscr{T}_{\tau(RN)}\} \cup \tag{6}$$
$$\{Q_{\tau_j}(\xi(t[\tau_i]), \xi(t[\tau_j])) \mid RN(t) \in \mathcal{A}, \tau_i \in \mathscr{T}_{\tau(RN)} \text{ and } \textsc{child}_{\mathscr{T}}(\tau_i, \tau_j)\} \cup \tag{7}$$
$$\{Q_o(o) \mid o \in \mathcal{O}\} \cup \tag{8}$$
$$\{Q_t(o_1) \mid t = \langle U_1{:}o_1, \ldots, U_n{:}o_n\rangle \text{ occurs in } \mathcal{A}\}. \tag{9}$$

Fig. 8. The mapping $\gamma(\mathcal{A})$

$\xi : T_{\mathcal{O}}(\mathcal{U}) \to \mathcal{O}_{\mathcal{ALCQI}}$, with $\mathcal{O}_{\mathcal{ALCQI}} = \mathcal{O} \cup \mathcal{O}^t$ being the set of individual names in $\gamma(\mathcal{KB})$, $\mathcal{O} \cap \mathcal{O}^t = \varnothing$ and

$$\xi(t) \;=\; \begin{cases} o \in \mathcal{O}, & \text{if } t = \langle U{:}o\rangle \\ o \in \mathcal{O}^t, & \text{otherwise.} \end{cases}$$

Following [10], the mapping $\gamma(\mathcal{A})$ in Fig. 8 introduces a new concept name Q_o for each individual $o \in \mathcal{O}$ and a new concept name Q_t for each relation instance t occurring in $\mathcal{A}$, with each Q_t restricted as follows:

$$Q_t \sqsubseteq \exists^{\leqslant 1}\big(\textsc{path}_{\mathscr{T}}(\tau(t), \{U_1\})^{\dagger}\big)^{-}.$$
$$\exists\big(\textsc{path}_{\mathscr{T}}(\tau(t), \{U_2\})^{\dagger}\big).Q_{o_2} \sqcap \ldots \sqcap \exists\big(\textsc{path}_{\mathscr{T}}(\tau(t), \{U_n\})^{\dagger}\big).Q_{o_n}$$

Intuitively, (6) and (7) reify each relation instance axiom occurring in $\mathcal{A}$ using the projection signature of the involved tuple itself. The Formulas (8) and (9) together with the axioms for concepts Q_t guarantee that there is exactly one $\mathcal{ALCQI}$ individual reifying a given tuple in a relation instance axiom. Clearly, the size of $\gamma(\mathcal{KB})$ is polynomial in the size of $\mathcal{KB}$ under the same coding of the numerical parameters.

We are now able to state our main technical result.

Theorem 4. *A $\mathcal{DLR}^{\pm}$ knowledge base $\mathcal{KB}$ is satisfiable iff the $\mathcal{ALCQI}$ knowledge base $\gamma(\mathcal{KB})$ is satisfiable.*

As a direct consequence of this theorem and the fact that $\mathcal{DLR}$ is a sublanguage of $\mathcal{DLR}^{\pm}$, we obtain the following corollary.

Corollary 5. *Reasoning in $\mathcal{DLR}^{\pm}$ is* ExpTime*-complete.*

6 Implementation of a $\mathcal{DLR}^{\pm}$ API

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

We have used the $\mathcal{DLR}^{\pm}$ API within a plugin for general ontology reasoning for conceptual design tools based on languages such as EER, UML (with OCL), and ORM (with derivation rules) [14]. This plugin supports the detection of inconsistencies, redundancies, complete derivations of the strictest implicit constructs and unexpected behaviours. Reasoning helps the modeller to detect relevant formal properties of the ontology that may be undetected during the modelling phase, which give rise to design quality degradation and/or increased development times and costs. The system is still at an early stage of completion, but it has been proved to be highly effective and efficient: indeed, it computes derivations in real time in the background while the ontology is being designed.

7 Conclusions

We have introduced the very expressive $\mathcal{DLR}^{+}$ description logic, which extends $\mathcal{DLR}$ with database oriented constraints. $\mathcal{DLR}^{+}$ is expressive enough to cover directly and more thoroughly the EER, UML, and ORM conceptual data models, among others. Although reasoning in $\mathcal{DLR}^{+}$ is undecidable, we show that a simple syntactic constraint on KBs restores decidability. In fact, the resulting logic $\mathcal{DLR}^{\pm}$ has the same complexity (EXPTIME-complete) as the basic $\mathcal{DLR}$ language. In other words, handling database constraints does not increase the complexity of reasoning in the logic. To enhance the use and adoption of $\mathcal{DLR}^{\pm}$, we have developed an API that fully implements reasoning for this language, and maps input knowledge bases into OWL. Using a standard OWL reasoner, we are able to provide a variety of $\mathcal{DLR}^{\pm}$ reasoning services.

We plan to investigate the problem of query answering under $\mathcal{DLR}^{\pm}$ ontologies and to check whether the complexity for this problem can be lifted from known results in $\mathcal{DLR}$ to $\mathcal{DLR}^{\pm}$.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., Zakharyaschev, M.: Reasoning over extended ER models. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 277–292. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75563-0_20

2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory Implementation and Applications. Cambridge University Press, New York (2003)
3. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artif. Intell. **168**(1–2), 70–118 (2005)
4. Calvanese, D., De Giacomo, G., Lenzerini, M.: Identification constraints and functional dependencies in description logics. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, pp. 155–160. Morgan Kaufmann (2001)
5. Calvanese, D., Fischl, W., Pichler, R., Sallinger, E., Simkus, M.: Capturing relational schemas and functional dependencies in RDFS. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI), pp. 1003–1011. AAAI Press (2014)
6. Calvanese, D., Giacomo, G.D., Lenzerini, M.: Conjunctive query containment and answering under description logic constraints. ACM Trans. Comput. Logic **9**(3), 22:1–22:31 (2008)
7. Chandra, A.K., Vardi, M.Y.: The implication problem for functional and inclusion dependencies is undecidable. SIAM J. Comput. **14**(3), 671–677 (1985)
8. Franconi, E., Mosca, A., Solomakhin, D.: ORM2: formalisation and encoding in OWL2. In: International Workshop on Fact-Oriented Modeling (ORM 2012), pp. 368–378 (2012)
9. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Morgan Kaufmann, San Francisco (2008)
10. Horrocks, I., Sattler, U., Tessaris, S., Tobies, S.: How to decide query containment under constraints using a description logic. In: Parigot, M., Voronkov, A. (eds.) LPAR 2000. LNAI, vol. 1955, pp. 326–343. Springer, Heidelberg (2000). doi:10.1007/3-540-44404-1_21
11. Kanellakis, P.C.: Elements of relational database theory. In: Meyer, A., Nivat, M., Paterson, M., Perrin, D., van Leeuwen, J. (eds.) The Handbook of Theoretical Computer Science, vol. B, Chap. 17, pp. 1075–1144. North Holland (1990)
12. Lukasiewicz, T., Cali, A., Gottlob, G.: A general datalog-based framework for tractable query answering over ontologies. Web Semant. Sci. Serv. Agents World Wide Web **14**, 57–83 (2012)
13. Patel-Schneider, P.F., Franconi, E.: Ontology constraints in incomplete and complete data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 444–459. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35176-1_28
14. Sportelli, F.: NORMA: A software for intelligent conceptual modeling. In: Proceedings of the Joint Ontology Workshops 2016 (JOWO-2016) (2016). http://ceurws.org/Vol-1660/demo-paper3.pdf
15. Sportelli, F., Franconi, E.: Formalisation of ORM derivation rules and their mapping into OWL. In: Debruyne, C., Panetto, H., Meersman, R., Dillon, T., Kühn, E., O'Sullivan, D., Ardagna, C.A. (eds.) OTM 2016. LNCS, vol. 10033, pp. 827–843. Springer, Heidelberg (2016)
16. Toman, D., Weddell, G.E.: Applications and extensions of PTIME description logics with functional constraints. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009, pp. 948–954 (2009)