Noname manuscript No.
(will be inserted by the editor)

Using Feature-Based Description Logics to avoid Duplicate Elimination in Object-Relational Query Languages

David Toman · Grant Weddell

Received: date / Accepted: date

Abstract A sound inference procedure is presented for removing operations that eliminate duplicates in queries formulated in a bag-algebra. The procedure is shown complete for positive queries over finite databases, and operates by appeal to logical consequence problems for feature-based description logics in which a TBox embeds an object-relational schema. For unions of conjunctive queries in which an embedded schema excludes cover constraints, the procedure runs in PTIME, and in EXPTIME otherwise.

1 Introduction

Query plans for queries in SQL are usefully abstracted as expressions in a bag-algebra, and are usually required to compute answers that are sets of tuples. Consequently, such plans may ultimately require operations that perform *tuple duplicate elimination* (TDE) in query answers. Such operations often constitute a considerable performance penalty, e.g., by inducing a temporary store and sorting of intermediate results that would otherwise be avoided by pipelining which often suffices for the remaining operations. In this paper, we show how dialects of the FunDL family of feature-based *description logics* (DLs) [9] can be usefully employed in rewrite rules that reduce the scope or eliminate the occurrence of TDE operations for a bag-algebra over object-relational data sources, thus controlling and possibly eliminating the need for unbounded store and for sorting in query evaluation.

We focus on two FunDL dialects with respective EXPTIME and PTIME complexity for their logical consequence problems: *partial-$\mathcal{DLFDE}$* and *partial-$\mathcal{CFD}_{nc}$*. In common with all FunDL dialects, these logics were designed for capturing and integrating data sources that include a variety of common equality and tuple generating dependencies. For example, either logic is capable of embedding in a so-called TBox the object-relational schema illustrated in Figure 1, presumably the metadata for a hypothetical university data source. Here: (1) single arrows are attributes, realized as *features* in *partial-$\mathcal{DLFDE}$* and *partial-$\mathcal{CFD}_{nc}$* (denoting partial unary functions), and (2) underlined attributes and double arrows indicate primary keys and inheritance respectively, and are embedded with so-called *path functional de-*

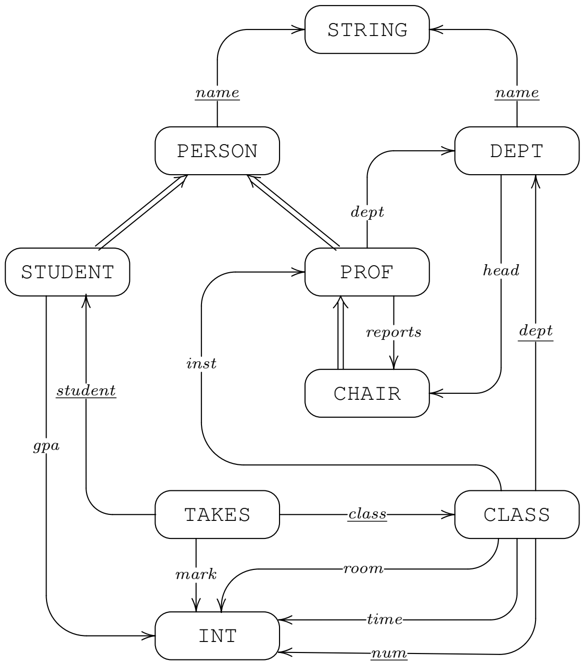

Fig. 1 A UNIV OBJECT-RELATIONAL SCHEMA.

D. Toman · G. Weddell
Cheriton School of Computer Science, University of Waterloo
200 University Ave. W., Waterloo, ON N2L3G1, Canada
Tel.: +1(519)888-4567
E-mail: {david,gweddell}@uwaterloo.ca

pendencies (PFDs), a concept constructor common to all member FunDL dialects that generalizes the notions of primary keys, uniqueness constraints and functional dependencies ubiquitous in object-relational schemata.

Figure 1 itself can be viewed as a purely relational schema with SQL tables. For example, consider the following SQL data definition for STUDENT and TAKES:

```
create table STUDENT (
  name STRING not null,
  gpa INT not null,
  primary key (name),
  foreign key (name)
  references PERSON (name) );

create table TAKES (
  class-dept-name STRING not null,
  class-num INT not null,
  student-name STRING not null,
  mark INT not null,
  primary key (class-dept-name,
    class-num, student-name),
  constraint student
    foreign key (student-name)
    references STUDENT (name),
  constraint class
    foreign key (class-dept-name, class-num)
    references CLASS (dept-name, num) );
```

Here, table columns correspond directly to attributes in Figure 1 that are integer or string valued, and indirectly to the remaining attributes in the figure according to primary key declarations. Indeed, this correspondence is sufficiently tight-knit that any need to manually provide, for example, so-called mapping rules in *ontology based data access* OBDA [2,10], is rendered mute [11].

To illustrate the utility of our results, consider the following queries over the UNIV schema expressed in terms of SQL/OQL-style syntax:

Example 1 "Find the names of students taking some course taught at the same time as some other course numbered $p1$ with an instructor in a department named $p2$":

```
(v1) select distinct s.name as n
     from STUDENT s, TAKES t, CLASS c
     where s = t.student and c.time = t.class.time
     and c.inst.dept.name = :p1 and c.num = :p2

(v2) select s.name as n
     from STUDENT s, TAKES t, CLASS c
     where s = t.student and c.time = t.class.time
     and c.inst.dept.name = :p1 and c.num = :p2
```

Note that $p1$ and $p2$ are parameters to the queries, and observe that the distinct keyword is absent in the second version. □

Example 2 "Find the names of all students and professors":

```
(v1) (select s.name as n from STUDENT s)
     union (select d.name as n from DEPT d)

(v1) (select s.name as n from STUDENT s)
     union all (select d.name as n from DEPT d)
```

Observe that the duplicate preserving union all operation in the second version replaces the TDE *union* operation in the first version. □

Our results will enable rewrite rules to infer that the second version of each query in the two examples must produce the same result as the first version. Indeed, all queries map in a straightforward way to formulations in our bag-algebra called $\mathcal{QL}$, and the $\mathcal{QL}$ formulations of the second versions are absent of $\mathcal{QL}$'s TDE operation.

The remainder of the paper is organized as follows. The necessary background and definitions are given next in Section 2. Here, we review the FunDL dialects of DLs, and introduce our bag-algebra, $\mathcal{QL}$. In Section 3, we consider base cases of TDE removal in $\mathcal{QL}$ that apply for CQs (conjunctive queries), and for UCQs (unions of CQs). Our main result is given in Section 4 which shows how the base cases can be generalized to cover all positive queries in $\mathcal{QL}$ via a notion of *query contexts*. In Section 5, we consider arbitrary $\mathcal{QL}$ queries with negation, and the issue of completeness for infinite interpretations. Summary comments and conclusions then follow in Section 6.

2 Background and Definitions

We begin by reviewing two FunDL dialects of DLs. Recall that such dialects are feature-based, that is, they substitute the usual notion of *roles* in DLs that are interpreted as binary predicates with features that are interpreted as partial functions. Also recall that all such dialects include the PFD concept constructor for capturing common varieties of equality generating dependencies in object-relational schema.

We begin by introducing TBoxes and the logical consequence problem for various FunDL dialects (see [9] for a more thorough review).

Definition 1 (Feature-Based Dialects of DLs)
Let F and PC be sets of feature names and primitive concept names, respectively. A *partial path expression* is defined by the grammar "$\mathsf{Pf} ::= f.\,\mathsf{Pf} \mid id$" for $f \in \mathsf{F}$. We define derived *concept descriptions* by the grammar on the left-hand-side of Figure 2. (The final production labelled PFD introduces *path functional dependency* concepts that enable capturing a variety of equality generating dependencies, see below.) An *inclusion dependency* $\mathcal{C}$ is an expression of the form $C_1 \sqsubseteq C_2$. A *terminology* (TBox) $\mathcal{T}$ consists of a finite set of inclusion dependencies. A *posed question* $\mathcal{Q}$ is a single inclusion dependency.

The *semantics* of expressions is defined with respect to a structure $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$, where $\triangle$ is a domain of "objects" and $\cdot^{\mathcal{I}}$ an interpretation function that fixes the interpretations of primitive concepts A to be subsets of $\triangle$ and primitive features f to be partial functions $f^{\mathcal{I}} : \triangle \to \triangle$. The interpretation is extended to partial path expressions, $id^{\mathcal{I}} = \lambda x.x$,

SYNTAX: C ::=	SEMANTICS: DEFN OF $(\cdot)^{\mathcal{I}}$	
A	$A^{\mathcal{I}} \subseteq \triangle$	(primitive concept; $A \in \mathsf{PC}$)
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	(conjunction)
$\forall \, \mathsf{Pf} \, .C$	$\{x \mid \mathsf{Pf}^{\mathcal{I}}(x) \in C^{\mathcal{I}}\}$	(value restriction)
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	(disjunction)
$\neg C$	$\triangle - C^{\mathcal{I}}$	(negation)
$\top$	$\triangle$	(top)
$\bot$	$\emptyset$	(bottom)
$(\mathsf{Pf}_1 = \mathsf{Pf}_2)$	$\{x \mid \mathsf{Pf}_1^{\mathcal{I}}(x) = \mathsf{Pf}_2^{\mathcal{I}}(x)\}$	(same-as)
$C : \{\mathsf{Pf}_1, ..., \mathsf{Pf}_k\} \to \mathsf{Pf}$	$\{x \mid \forall y \in (C \sqcap \exists \mathsf{Pf})^{\mathcal{I}}.(x \in (\exists \mathsf{Pf})^{\mathcal{I}} \wedge \bigwedge_{i=1}^{k} (\mathsf{Pf}_i^{\mathcal{I}}(x) = \mathsf{Pf}_i^{\mathcal{I}}(y)) \Rightarrow \mathsf{Pf}^{\mathcal{I}}(x) = \mathsf{Pf}^{\mathcal{I}}(y))\}$	(PFD)

Fig. 2 CONCEPT CONSTRUCTORS IN *partial-$\mathcal{DLFDE}$*.

(*attribute typing*)	PERSON $\sqsubseteq \forall name.$STRING
(*unary primary keys*)	PERSON $\sqsubseteq$ PERSON : $\{name\} \to id$
(*inheritance*)	STUDENT $\sqsubseteq$ PERSON
(*binary primary keys*)	CLASS $\sqsubseteq$ CLASS : $\{dept, num\} \to id$

(a) ATTRIBUTES, PRIMARY KEYS AND INHERITANCE.

(*disjoint classes*)	PERSON $\sqsubseteq \neg$DEPT
(*disjoint attribute values*)	PERSON $\sqsubseteq$ DEPT : $\{name\} \to id$
(*views*)	$\forall reports.$CHAIR $\sqsubseteq$ PROF
(*time/space interactions*)	TAKES $\sqsubseteq$ TAKES : $\{student, class.room, class.time\} \to class$
(*cover*)	PERSON $\sqsubseteq$ STUDENT $\sqcup$ PROF
(*inapplicable attributes*)	DEPT $\sqsubseteq \neg\exists gpa$

(b) ADDITIONAL DATA DEPENDENCIES.

Fig. 3 PART OF A $\mathcal{T}_{\text{UNIV}}$ TBOX EMBEDDING THE UNIV SCHEMA.

$(f.\,\mathsf{Pf})^{\mathcal{I}} = \mathsf{Pf}^{\mathcal{I}} \circ f^{\mathcal{I}}$, in the natural way, and to derived concept descriptions C as defined in the centre column of Figure 2. Also, to improve readability, we omit mention of trailing occurrences of id in partial path expressions.

A *strict* interpretation of undefined values is assumed, which means that argument terms *must* be defined whenever equality and set membership do hold. This implies the interpretation of any value restriction $\forall \, \mathsf{Pf} \, .C$ will *exclude* any object for which Pf is undefined. To denote all objects for which Pf *is* defined we write $\exists \mathsf{Pf}$ as shorthand for $\forall \, \mathsf{Pf} \, .\top$.

An interpretation $\mathcal{I}$ *satisfies an inclusion dependency* $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$ and is a *model of* $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if it satisfies all inclusion dependencies in $\mathcal{T}$. A *logical consequence problem* is to determine if $\mathcal{T} \models \mathcal{Q}$ holds for a given $\mathcal{T}$ and $\mathcal{Q}$, that is, if $\mathcal{Q}$ is satisfied in all models of $\mathcal{T}$. □

As defined, this logical consequence problem is undecidable for a variety of reasons. Thus, all member dialects of the FunDL family limit how concepts may be used in inclusion dependencies and in posed questions. The following defines a representative pair of dialects reviewed in [9] with respective complexity for their logical consequence problems in EXPTIME and PTIME.

Definition 2 (*partial-$\mathcal{DLFDE}$* and *partial-$\mathcal{CFD}_{nc}$*) Inclusion dependencies and posed questions in both dialects must adhere to the respective forms $C \sqsubseteq D$ and $E \sqsubseteq E$, where, for *partial-$\mathcal{DLFDE}$*, concepts C, D and E are given by the grammar:

$$\begin{array}{rl} C ::= & A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall f.C \mid \neg C \mid \top \\ D ::= & C \mid D_1 \sqcap D_2 \mid D_1 \sqcup D_2 \mid \forall f.D \\ \mid & C : \{\mathsf{Pf}_1, ..., \mathsf{Pf}_k\} \to \mathsf{Pf} \\ E ::= & C \mid \bot \mid E_1 \sqcap E_2 \mid \neg E \mid \forall f.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2); \end{array}$$

and, for *partial-$\mathcal{CFD}_{nc}$*, by the grammar:

$$\begin{array}{rl} C ::= & A \mid \forall f.C \\ D ::= & C \mid \neg C \mid D_1 \sqcap D_2 \mid \forall f.D \mid C : \{\mathsf{Pf}_1, ..., \mathsf{Pf}_k\} \to \mathsf{Pf} \\ E ::= & C \mid \bot \mid E_1 \sqcap E_2 \mid \forall f.E \mid (\mathsf{Pf}_1 = \mathsf{Pf}_2). \end{array}$$

In addition, in both cases the PFD constructor must adhere to a *boundary* condition, in particular, to have either of the following two forms:

$$\begin{array}{l} C : \{\mathsf{Pf}_1, \ldots, \mathsf{Pf}.\,\mathsf{Pf}_i, \ldots, \mathsf{Pf}_k\} \to \mathsf{Pf}\text{, and} \\ C : \{\mathsf{Pf}_1, \ldots, \mathsf{Pf}.\,\mathsf{Pf}_i, \ldots, \mathsf{Pf}_k\} \to \mathsf{Pf}\,.f, \end{array}$$

for some feature f. □

The logics can also accommodate posed questions of the form $E \sqsubseteq E : \mathsf{Pf}_1, \ldots, \mathsf{Pf}_k \to \mathsf{Pf}$ by converting then to a subsumption between plain equational concepts [6]. To illustrate how these dialects can be used to capture object-relational schemata, again consider the UNIV data source depicted in Figure 1. Figure 3 (a) is a list of representative examples of the inclusion dependencies in a TBox that embeds UNIV, that is, for capturing attributes, primary keys and inheritance. Note that an exhaustive list would include, for example, thirteen additional inclusion dependencies to capture attribute typing alone. Figure 3 (b) is a list of additional data dependencies beyond what is indicated by the UNIV schema that would be satisfied by any UNIV database, expressing time/space dependencies via PFDs for example. In this case, an exhaustive list would, for example, include seventeen additional inclusion dependencies expressing disjointness between classes. Note that all qualify as *partial-$\mathcal{DLFDE}$* inclusion dependencies, and that all but the final two dependencies in Figure 3 (b) qualify as *partial-$\mathcal{CFD}_{nc}$* inclusion dependencies.

We now introduce the query language $\mathcal{QL}$ in which a query denotes a function that, when applied to an interpretation, obtains a *bag of tuples*. Here, tuples are semantic artifacts that will correspond to finite valuations, a circumstance that suffices for capturing when rewrite rules over $\mathcal{QL}$ should be viewed as correct.

Definition 3 (Query Language $\mathcal{QL}$) Let F and PC be respective sets of feature names and primitive concept names, as with feature-based DLs, and let V be a set of variable

SYNTAX: Q ::=	SEMANTICS: DEFN OF $[\![Q]\!](\mathcal{I})$ †
$A(x)$	$\{\{x \mapsto e\} : 1 \mid e \in A^{\mathcal{I}}\}$
$x_1.\mathsf{Pf}_1 = x_2.\mathsf{Pf}_2$	$\{t : 1 \mid \mathsf{Pf}_1^{\mathcal{I}}(t(x_1)) = \mathsf{Pf}_2^{\mathcal{I}}(t(x_2))\}$
$Q_1 \wedge Q_2$	$\{t : n_1 \cdot n_2 \mid \bigwedge_{i\in\{1,2\}} t[\mathsf{Fv}(Q_i)] : n_i \in [\![Q_i]\!](\mathcal{I})\}$
$Q_1 \vee Q_2$	$\{t : n_1 + n_2 \mid \bigwedge_{i\in\{1,2\}} t[\mathsf{Fv}(Q_i)] : n_i \in [\![Q_i]\!](\mathcal{I})\}$ $\cup \{t : n \mid t[\mathsf{Fv}(Q_1)] : n \in [\![Q_1 \wedge \neg Q_2]\!](\mathcal{I})\}$ $\cup \{t : n \mid t[\mathsf{Fv}(Q_2)] : n \in [\![Q_2 \wedge \neg Q_1]\!](\mathcal{I})\}$
$\exists x.Q$	$\{t : \sum_{t' : n \in S_t} n \mid S_t \neq \emptyset\}$ where $S_t = \{t' : m \mid t' : m \in [\![Q]\!](\mathcal{I}) \wedge t'[\mathsf{Fv}(\exists x.Q)] = t\}$
$\{Q\}$	$\{t : 1 \mid t : n \in [\![Q]\!](\mathcal{I})\}$
$\neg Q$	$\{t : 1 \mid \neg\exists n \text{ s.t. } t : n \in [\![Q]\!](\mathcal{I})\}$

† where t denotes a partial valuation with domain $\mathsf{Fv}(Q)$.

Fig. 4 SYNTAX AND SEMANTICS OF $\mathcal{QL}$.

names disjoint from F and PC. We define *queries* according to the grammar on the left-hand-side of Figure 4 in which non-logical parameters are given by unary predicates A in PC, by path expressions Pf_i over F, and by variables x_i in V. Without loss of generality, queries must also satisfy a *single use* condition for variables, in particular, where no variable is existentially quantified more than once. Also, to improve readability for occurrences of equality in queries, we omit mention of id in terms of the form "$x.\,id$".

The semantics for queries is also defined with respect to a struct $\mathcal{I} = (\triangle, \cdot^{\mathcal{I}})$, and is given on the right-hand-side of Figure 4. Here, a bag semantics for a query Q is assumed in which $[\![Q]\!](\mathcal{I})$, the function denoted by Q applied to $\mathcal{I}$, computes a possibly infinite set of pairs $\langle t, n\rangle$, written "$t : n$", in which tuple t is a finite partial valuation over $\mathsf{Fv}(Q)$, the free variables of Q, to objects in $\triangle$, and in which n denotes a cardinal number.[1] Also, it holds that $t_1 \neq t_2$ whenever $\{t_1 : n_1, t_2 : n_2\} \subseteq [\![Q]\!](\mathcal{I})$. In defining the semantics, we write $t[V]$ to denote the partial valuation obtained by restricting the domain of t to the set V of variables.

A query Q is *range restricted* if it satisfies additional syntactic conditions. First, for existential quantification and disjunction, $x \in \mathsf{Fv}(Q)$ and $\mathsf{Fv}(Q_2) = \mathsf{Fv}(Q_1)$ hold respectively. And second, Q is among the queries defined by a modification to the grammar on the left-hand-side of Figure 4 in which the productions for equality and for negation are replaced by the following respective range restricted variants: $Q \wedge x_1.\mathsf{Pf}_1 = x_2.\mathsf{Pf}_2$ and $Q_1 \wedge \neg Q_2$, where now also $\mathsf{Fv}(Q_2) \subseteq \mathsf{Fv}(Q_1)$ must hold with negation, $x_1 \in \mathsf{Fv}(Q)$ must hold with equality, and, if $\mathsf{Pf}_2 \neq id$, $x_2 \in \mathsf{Fv}(Q)$ must also hold with equality.

[1] Thus, arithmetic operations on n denote cardinal arithmetic.

Finally, for a given TBox $\mathcal{T}$, we write $Q_1 \overset{\mathcal{T}}{=} Q_2$ to denote a *rewrite rule*, and say that the rule is *correct* when

$$[\![Q_1]\!](\mathcal{I}) = [\![Q_2]\!](\mathcal{I})$$

holds for any structure $\mathcal{I}$ for which $\mathcal{I} \models \mathcal{T}$ and $n < \aleph_0$ whenever $t : n \in [\![Q_1]\!](\mathcal{I})$. □

Note that a TBox capturing an object-relational data source will have a finite structure, as in the case of the above UNIV schema, since databases that underlie the tables and classes in these data sources will themselves be finite. Observe in such cases that the semantics of range restricted $\mathcal{QL}$ also qualifies as an operational semantics that aligns with the relational algebra extended the usual way with bag semantics. Indeed, query plans and bag-algebras mentioned in our introductory comments naturally correspond to queries in range restricted $\mathcal{QL}$, and TDE operators to occurrences of $\mathcal{QL}$'s "$\{\cdot\}$" construct.

Example 3 Consider the first version of the queries in Examples 1 and 2 above. Translating these queries into respective formulations in $\mathcal{QL}$ is straightforward, and would obtain the following:

$$\begin{aligned}&\{\exists s.\exists t.\exists c.(\mathtt{STUDENT}(s) \wedge \mathtt{TAKES}(t) \wedge \mathtt{CLASS}(c)\\ &\quad \wedge\ s.name = n\\ &\quad \wedge\ s = t.student \wedge c.time = t.class.time\\ &\quad \wedge\ c.inst.dept.name = p1 \wedge c.num = p2\},\ \text{and}\end{aligned}$$

$$\begin{aligned}&\{(\exists s.(\mathtt{STUDENT}(s) \wedge s.name = n)\\ &\quad \vee (\exists d.(\mathtt{DEPT}(d) \wedge d.name = n)\}.\end{aligned}$$

Observe that n is the *answer* variable for both queries, and also that we have followed our protocol of removing mention of id in terms such as "$n.\,id$". □

Observe how both formulations are also range restricted, and that the range restricted queries will qualify as query plans in the context of a UNIV TBox. To reiterate our introductory comments, rewrite rules introduced below can eliminate $\mathcal{QL}$'s TDE operator "$\{\cdot\}$" in these queries via logical consequence problems for the UNIV TBox that includes the dependencies listed in Figure 3. Indeed, doing so would obtain the respective second versions of these queries in Examples 1 and 2.

3 Duplicate Elimination Removal: the Base Cases

In this section, we consider how $\mathcal{QL}$'s TDE operator can be moved within the scope of both existential quantification and disjunction when occurring near the root of a query. In preparation, we introduce an abstraction of positive $\mathcal{QL}$ queries using concepts in the description logic $\mathcal{DL}$ as follows:

Definition 4 (Positive Queries as FunDL concepts)
Let Q be a positive $\mathcal{QL}$ query satisfying the single use condition in which quantified variables have unique names. We

define a derived FunDL concept E_Q induced by Q according to the following:

$$\begin{array}{ll} E_{A(x)} & = \forall f_x.A, \\ E_{x.\,\mathsf{Pf}_1 = y.\,\mathsf{Pf}_2} & = (f_x.\,\mathsf{Pf}_1 = f_y.\,\mathsf{Pf}_2), \\ E_{Q_1 \wedge Q_2} & = E_{Q_1} \sqcap E_{Q_2}, \\ E_{Q_1 \vee Q_2} & = E_{Q_1} \sqcup E_{Q_2}, \\ E_{\exists x.Q} & = E_Q, \text{and} \\ E_{\{Q\}} & = E_Q, \end{array}$$

where f_x are distinct fresh features not appearing in $\mathcal{T}$ that are associated with variables V. □

Although features are interpreted as partial functions in both *partial-$\mathcal{DLFDE}$* and *partial-$\mathcal{CFD}_{nc}$*, the strict interpretation of equality together with the first two cases in the above ensures that the f_x features will be defined for all variables of Q.

We use this abstraction primarily to reason about duplicate semantics of $\mathcal{QL}$ queries. However, to illustrate its utility, it can also be shown to correctly abstract the *query emptiness* problem (for positive $\mathcal{QL}$ queries):

Theorem 1 *Let $\mathcal{T}$ be a TBox that represents a database schema and Q a positive query. Then $[\![Q]\!](\mathcal{I})$ is empty in every model $\mathcal{I}$ of $\mathcal{T}$ if and only if $\mathcal{T} \models E_Q \sqsubseteq \bot$.*

Proof (by contradiction) Let $\mathcal{I}$ be a model of $\mathcal{T}$ such that there is an object $o \in E_Q^{\mathcal{I}}$. Then, by simultaneous induction on the structure of Q and E_Q we have

Case $A(x)$: $o \in (\forall f_x.A)^{\mathcal{I}}$ and thus $f_x^{\mathcal{I}}(o)$ (that must exist by the definition of value restrictions) qualifies as an answer $\{x \mapsto f_x^{\mathcal{I}}(o)\} : 1 \in [\![Q]\!](\mathcal{I})$;

Case $x.\,\mathsf{Pf}_1 = y.\,\mathsf{Pf}_2$: $o \in (f_x.\,\mathsf{Pf}_1 = f_y.\,\mathsf{Pf}_2)^{\mathcal{I}}$ and thus $\{x \mapsto f_x^{\mathcal{I}}(o), y \mapsto f_y^{\mathcal{I}}(o)\} : 1 \in [\![Q]\!](\mathcal{I})$;

Case $Q_1 \wedge Q_2$: $o \in (E_{Q_1} \sqcap E_{Q_2})^{\mathcal{I}}$ and, by the definition of $\sqcap$ and the induction hypothesis we must have a partial valuation $t = \{x_i \mapsto f_{x_i}^{\mathcal{I}}(o) \mid x_i \in \mathsf{Fv}(Q_1 \wedge Q_2)\}$ such that $t[\mathsf{Fv}(Q_1)] : n_1 \in [\![Q_1]\!](\mathcal{I})$ and $t[\mathsf{Fv}(Q_2)] : n_2 \in [\![Q_2]\!](\mathcal{I})$ and $t[\mathsf{Fv}(Q_1)]$ and $t[\mathsf{Fv}(Q_2)]$ must agree on the values of common variables due to the definition of $E_{Q_1 \wedge Q_2}$ with $n_1, n_2 > 0$. Hence $t : n_1 \cdot n_2 \in [\![Q_1 \wedge Q_2]\!](\mathcal{I})$;

Case $Q_1 \vee Q_2$: $o \in (E_{Q_1} \sqcup E_{Q_2})^{\mathcal{I}}$. Hence, by the inductive hypothesis, there must exist a tuple $t = \{x_i \mapsto f_{x_i}^{\mathcal{I}}(o) \mid x_i \in \mathsf{Fv}(Q_1 \vee Q_2)\}$ such that $t[\mathsf{Fv}(Q_1)] : n \in [\![Q_1]\!](\mathcal{I})$ or $t[\mathsf{Fv}(Q_2)] : n \in [\![Q_2]\!](\mathcal{I})$ for some $n > 0$. Thus $t : m \in [\![Q_1 \vee Q_2]\!](\mathcal{I})$ for some $m \geq n$.

Case $\exists x.Q$: $o \in (E_{\exists x.Q})^{\mathcal{I}} = E_Q^{\mathcal{I}}$. By inductive hypothesis we have at least one $t : n \in [\![Q]\!](\mathcal{I})$ where $n > 0$ and, by the definition of the semantics of $\exists x.Q$ we have $t[\mathsf{Fv}(\exists x.Q)] : m \in [\![\exists x.Q]\!](\mathcal{I})$ for some $m \geq n$.

Case $\{Q\}$: $o \in E_{\{Q\}}^{\mathcal{I}} = E_Q^{\mathcal{I}}$ and thus there must be $t : n \in [\![Q]\!](\mathcal{I})$ with $n > 0$. Hence, $t : 1 \in [\![\{Q\}]\!](\mathcal{I})$.

In all cases we constructed at least one answer valuation for the query in question starting from $o \in E_Q^{\mathcal{I}}$, contradiction with Q being empty. The reverse direction follows since all cases above are "if and only if" cases in the following sense: at the end of each case we extend the model $\mathcal{I}$ of $\mathcal{T}$ that makes the query Q non-empty by an additional object o such that $o \in E_Q^{\mathcal{I}}$, and where interpretations of features $f_{y_i}^{\mathcal{I}}(o) = w_i$ for $y_i \mapsto w_i \in t$ and $y_i \in \mathsf{Fv}(Q)$. It is easy to check that this interpretation is still a model of $\mathcal{T}$ in which $E_Q^{\mathcal{I}}$ is non-empty. □

3.1 Conjunctive Queries and Projections

We now consider applying the above abstraction to manipulating $\mathcal{QL}$'s TDE operator, starting with CQs:

Theorem 2 *Let $\mathcal{T}$ be TBox that represents a database schema and Q a $\mathcal{QL}$ query of the form*

$$Q_1 \wedge \{\exists x.Q_2\}, \tag{1}$$

where each Q_i is a CQ. Then the following rewrite rule is correct if and only if $\mathcal{T} \models E_Q \sqsubseteq E_Q : \mathsf{Fv}(Q) \to x$:

$$Q_1 \wedge \{\exists x.Q_2\} \overset{\mathcal{T}}{=} Q_1 \wedge \exists x.\{Q_2\}.$$

Proof Assume that $\mathcal{T} \not\models E_Q \sqsubseteq E_Q : \mathsf{Fv}(Q) \to x$. Then there must be a model $\mathcal{I}$ of $\mathcal{T}$ and two objects $o_1, o_2 \in \triangle$ such that $f_y^{\mathcal{I}}(o_1) = f_y^{\mathcal{I}}(o_2)$ for all $y \in \mathsf{Fv}(Q)$ and $f_x^{\mathcal{I}}(o_1) \neq f_x^{\mathcal{I}}(o_2)$. We construct partial valuations $t_2^1 = \{y_i \mapsto f_{y_i}^{\mathcal{I}}(o_1) \mid y_i \in \mathsf{Fv}(Q_2)\}$ $t_2^2 = \{y_i \mapsto f_{y_i}^{\mathcal{I}}(o_2) \mid y_i \in \mathsf{Fv}(Q_2)\}$ $t_1 = \{y_i \mapsto f_{y_i}^{\mathcal{I}}(o_1) \mid y_i \in \mathsf{Fv}(Q_1)\}$ such that $t_1 : n \in [\![Q_1]\!](\mathcal{I})$ and $t_2^1 : m_1, t_2^2 : m_2 \in [\![Q_2]\!](\mathcal{I})$ for $n, m_1, m_2 > 0$ as in Theorem 1. Then $t_2^1 : 1, t_2^2 : 1 \in [\![\{Q_2\}]\!](\mathcal{I})$ Then, however, $t_2^1[\mathsf{Fv}(Q_2) - \{x\}] : 1 \in [\![\{\exists x.Q_2\}]\!](\mathcal{I})$ while $t_2^1[\mathsf{Fv}(Q_2) - \{x\}] : m \in [\![\exists x.\{Q_2\}]\!](\mathcal{I})$ for $m \geq 2$ as $t_2^1[\mathsf{Fv}(Q_2) - \{x\}] = t_2^2[\mathsf{Fv}(Q_2) - \{x\}]$. Hence $Q_1 \wedge \{\exists x.Q_2\}$ will have n duplicate answers while $Q_1 \wedge \exists x.\{Q_2\}$ will have at least $2n$ duplicate answers (constructed out of t_1 and $t_2^1[\mathsf{Fv}(Q_2) - \{x\}]$).

For the other direction, if

$$[\![Q_1 \wedge \{\exists x.Q_2\}]\!](\mathcal{I}) \neq [\![Q_1 \wedge \exists x.\{Q_2\}]\!](\mathcal{I}),$$

there must be t such that $t : n \in [\![Q_1 \wedge \{\exists x.Q_2\}]\!](\mathcal{I})$ and $t : m \in [\![Q_1 \wedge \exists x.\{Q_2\}]\!](\mathcal{I})$ with $m \neq n$ (it is easy to see that the answers to these two queries only differ in the number of duplicates). For this to be possible, $t[\mathsf{Fv}(\exists x.\{Q_2\}) : m_2 \in [\![\exists x.\{Q_2\}]\!](\mathcal{I})$ for $m_2 \geq 2$ (while $t[\mathsf{Fv}(\{\exists x.Q_2\}) : 1 \in [\![\{\exists x.Q_2\}]\!](\mathcal{I})$ by the definition of the duplicate elimination operator). Hence there must be at least two distinct values v_1, v_2 binding to x in Q_2 (extending t). We now extend $\mathcal{I}$ with two new objects o_1, o_2 such that $o_1, o_2 \in E_Q^{\mathcal{I}}$ and where interpretations of features $f_x^{\mathcal{I}}(o_1) = v_1$, $f_x^{\mathcal{I}}(o_2) = v_2$, and $f_{y_i}^{\mathcal{I}}(o_1) = f_{y_i}^{\mathcal{I}}(o_2) = w_i$ for $y_i \mapsto w_i \in t$ and $y_i \in \mathsf{Fv}(Q)$. It is easy to check that this is a model of $\mathcal{T}$ that violates $E_Q \sqsubseteq E_Q : \mathsf{Fv}(Q) \to x$. □

As presented, the above rewrite rule has limited utility on its own. (Theorem 2, however, is used in proving our main result in the next section.) This utility can be improved by admitting a $\mathcal{QL}$ query of the form "true" (denoting a function returning $\{\{\} : 1\}$), by introducing the rewrite rule

$$Q \overset{\mathcal{T}}{=} (\mathsf{true} \land Q), \tag{2}$$

and by introducing more general rewrite rules of the form (for $i > 0$):

$$\begin{aligned} &Q_1 \land \exists x_1. \cdots .\exists x_i.\{\exists x.Q_2\} \\ &\overset{\mathcal{T}}{=} \; Q_1 \land \exists x_1. \cdots .\exists x_i.\exists x.\{Q_2\} \end{aligned} \tag{3}$$

when the condition

$$\mathcal{T} \models E_Q \sqsubseteq E_Q : \mathsf{Fv}(Q) \cup \{x_1, \ldots, x_i\} \to x$$

holds. These additional rules are straightforward corollaries of Theorem 2. Altogether, this expanded set of rules can be usefully applied to the first of our running example UNIV queries:

Example 4 Consider the first $\mathcal{QL}$ query in Example 3. Applying the above rewrite rules (1), (2) and (3), would obtain the following equivalent formulation:

$$\begin{aligned} &\exists s.\exists t.\exists c.\{\mathtt{STUDENT}(s) \land \mathtt{TAKES}(t) \land \mathtt{CLASS}(c) \\ &\quad \land\ s.name = n \\ &\quad \land\ s = t.student \land c.time = t.class.time \\ &\quad \land\ c.inst.dept.name = p1 \land c.num = p2\}. \end{aligned}$$

To qualify as correct, the final application of one of the (3) rules, for example, requires ensuring that

$$\mathcal{T}_{\text{UNIV}} \models E_Q \sqsubseteq E_Q : \{p1, p2, s, t\} \to c$$

holds, where E_Q is the concept

$$\begin{aligned} &(\forall f_s.\mathtt{STUDENT} \land \forall f_t.\mathtt{TAKES} \land \forall f_c.\mathtt{CLASS} \\ &\quad \land\ f_s.name = f_n \\ &\quad \land\ f_s = f_t.student \land f_c.time = f_t.class.time \\ &\quad \land\ f_c.inst.dept.name = p1 \land f_c.num = p2). \end{aligned}$$ □

The rules presented so far derive from earlier work [7] in which conjunctive queries were first mapped to a normal form. In Section 4 that follows, we introduce what we call query contexts to mitigate any need to "migrate" queries to normal forms, to introduce additional operations in $\mathcal{QL}$ that are syntactic sugar, or to introduce an unbounded number of rewrite rules.

3.2 Unions of Conjunctive Queries

As with existential quantification (or projection), $\mathcal{QL}$'s TDE operator does not freely commute with disjunction. The following theorem provides a sufficient and, for range-restricted queries, necessary condition in which such commuting constitutes a correct rewriting:

Theorem 3 *Let $\mathcal{T}$ be a TBox that represents a database schema and consider a $\mathcal{QL}$ query of the form*

$$Q \land \{Q_1 \lor Q_2\}. \tag{4}$$

Then the following rewrite rule is correct if and only if $\mathcal{T} \models E_Q \sqcap E_{Q_1} \sqcap E_{Q_2} \sqsubseteq \bot$.

$$Q \land \{Q_1 \lor Q_2\} \overset{\mathcal{T}}{=} Q \land (\{Q_1\} \lor \{Q_2\}).$$

Proof Assume that $\mathcal{T} \not\models E_Q \sqcap E_{Q_1} \sqcap E_{Q_2} \sqsubseteq \bot$. Then there must be an model $\mathcal{I}$ of $\mathcal{T}$ and an object o such that $o \in Q^{\mathcal{I}} \cap E^{\mathcal{I}}_{Q_1} \cap E^{\mathcal{I}}_{Q_2}$. Then there must be t such that

$$\begin{aligned} &t[\mathsf{Fv}(Q)], m \in [\![Q]\!](\mathcal{I}), \\ &t[\mathsf{Fv}(Q_1)], n_1 \in [\![Q_1]\!](\mathcal{I}), \textit{ and} \\ &t[\mathsf{Fv}(Q_2)], n_2 \in [\![Q_2]\!](\mathcal{I}) \end{aligned}$$

as in Theorem 1. Then, however

$$\begin{aligned} &t[\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q_2)] : 1 \in [\![\{Q_1 \lor Q_2\}]\!](\mathcal{I}) \textit{ and} \\ &t[\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q_2)] : 2 \in [\![\{Q_1\} \lor \{Q_2\}]\!](\mathcal{I}) \end{aligned}$$

by the definition of $\mathcal{QL}$ and thus $t : m \in [\![Q \land \{Q_1 \lor Q_2\}]\!](\mathcal{I})$ but $t : 2m \in [\![Q \land (\{Q_1\} \lor \{Q_2\})]\!](\mathcal{I})$. Conversely, if there is a t such that $t[\mathsf{Fv}(Q)], m \in [\![Q]\!](\mathcal{I})$ and

$$\begin{aligned} &t[\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q_2)] : 1 \in [\![\{Q_1 \lor Q_2\}]\!](\mathcal{I}) \textit{ while} \\ &t[\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q_2)] : 2 \in [\![\{Q_1\} \lor \{Q_2\}]\!](\mathcal{I}) \end{aligned}$$

(this is the only possible case since, again, the two queries can only differ in the numbers of duplicates), then for the later to happen, $t[\mathsf{Fv}(Q_1)] : 1 \in [\![\{Q_1\}]\!](\mathcal{I})$ and $t[\mathsf{Fv}(Q_2)] : 1 \in [\![\{Q_2\}]\!](\mathcal{I})$ and thus $t[\mathsf{Fv}(Q_1)] : n \in [\![Q_1]\!](\mathcal{I})$ and $t[\mathsf{Fv}(Q_2)] : m \in [\![Q_2]\!](\mathcal{I})$ for $m, n > 0$. We now extend the model $\mathcal{I}$ of $\mathcal{T}$ by an object o such that $o \in (E_{Q_1} \sqcap E_{Q_2})^{\mathcal{I}}$ and where interpretations of features $f^{\mathcal{I}}_{y_i}(o) = w_i$ for $y_i \mapsto w_i \in t$ and $y_i \in \mathsf{Fv}(Q)$. It is again easy to check that this is a model of $\mathcal{T}$ that violates $E_{Q_1} \sqcap E_{Q_2} \sqsubseteq \bot$. □

Note again that the utility of the theorem can be improved by admitting "true" queries in $\mathcal{QL}$ and rewrite rule (2) above.

Example 5 Consider the second $\mathcal{QL}$ query in Example 3. Applying the above rewrite rule (4) together with (2) would obtain the following equivalent formulation:

$$\begin{aligned} &\{\exists s.(\mathtt{STUDENT}(s) \land s.name = n\} \\ &\qquad \lor\ \{\exists d.(\mathtt{DEPT}(d) \land d.name = n\}. \end{aligned}$$

Again, to qualify as correct, the application of rule (4) requires ensuring that $\mathcal{T}_{\text{UNIV}} \models E_{\mathsf{true}} \land E_{Q_1} \land E_{Q_2} \sqsubseteq \bot$ holds, where E_{true}, E_{Q_1} and E_{Q_2} are the respective concepts $\top$,

$$\begin{aligned} &\forall f_s.\mathtt{STUDENT} \land f_s.name = f_n, \text{ and} \\ &\forall f_d.\mathtt{DEPT} \land f_s.name = f_n. \end{aligned}$$ □

The rules we have introduced in Theorems 2 and 3 are essentially all that is needed to exhaustively determine when $\mathcal{QL}$'s TDE operator can be eliminated in UCQs. An algorithm to do this begins by first pushing duplicate elimination operators through the top-level disjunctions, and then through existential quantifiers in the conjunctive subqueries. This yields a PTIME algorithm in the size of Q since, in the worst case, we need to use Theorem 3 for every pair of conjunctive subqueries and then Theorem 2 for every existential quantifier. Indeed, the rewriting rules do not change the size of the query.

4 Positive Queries and Query Contexts

We now consider general positive queries in $\mathcal{QL}$. One approach would be to convert a given query to a UCQ first, and then apply the algorithm outlined just above. However, the conversion to an equivalent UCQ query can yield a reformulation that is exponential in the size of the original query. In this section, we present a much cleaner approach that avoids this blowup. In particular, we introduce a notion of *query contexts* that allows alternative formulations of queries in $\mathcal{QL}$ in which rewrite rules can then refer to arbitrary subqueries via such query contexts.

Definition 5 (Query Contexts) A *query context* $Q[\cdot]$ is an arbitrary query in $\mathcal{QL}$ with a single occurrence of a *placeholder* "$[\cdot]$" occurring as a subquery. For given query contexts $Q_1[\cdot]$ and $Q_2[\cdot]$ and a given query Q, we write $Q_1[Q_2[\cdot]]$ (resp. $Q_1[Q]$) to denote a query context (resp. query) in which $[\cdot]$ has been substituted with $Q_2[\cdot]$ (resp. Q).

Thus, an equivalent formulation of any query Q containing a subquery Q' in $\mathcal{QL}$ would be $Q[Q']$, where $Q[\cdot]$ is the query context obtained from Q by replacing its subquery Q' with a placeholder.

Let $Q[\cdot]$ be a query context with Q a positive query. We define $C_{Q[\cdot]}$, a FunDL concept derived from $Q[\cdot]$, and the set of *context variables* $\mathsf{Fv}(Q[\cdot])$ of $Q[\cdot]$ as follows:

Context $Q[\cdot]$	Concept $C_{Q[\cdot]}$	Context Vars $\mathsf{Fv}(Q[\cdot])$
$[\cdot]$	$\top$	$\emptyset$
$Q[Q_1 \wedge [\cdot]]$	$E_{Q_1} \sqcap C_{Q[\cdot]}$	$\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q[\cdot])$
$Q[[\cdot] \wedge Q_1]$	$E_{Q_1} \sqcap C_{Q[\cdot]}$	$\mathsf{Fv}(Q_1) \cup \mathsf{Fv}(Q[\cdot])$
$Q[Q_1 \vee [\cdot]]$	$C_{Q[\cdot]}$	$\mathsf{Fv}(Q[\cdot])$
$Q[[\cdot] \vee Q_1]$	$C_{Q[\cdot]}$	$\mathsf{Fv}(Q[\cdot])$
$Q[\exists x.[\cdot]]$	$C_{Q[\cdot]}$	$\mathsf{Fv}(Q[\cdot])$
$Q[\{[\cdot]\}]$	$C_{Q[\cdot]}$	$\mathsf{Fv}(Q[\cdot])$

□

With contexts, we can now reformulate and extend the rewrite rules for manipulating the duplicate elimination operator as follows:

Theorem 4 *Let $\mathcal{T}$ be TBox that represents a database schema and Q a positive query. Then the following rewrite rules are correct:*

(1) $Q[\{[\{Q_1\}]\}] \overset{\mathcal{T}}{=} Q[\{[Q_1]\}]$
(2) $Q[\{A(x)\}] \overset{\mathcal{T}}{=} Q[A(x)]$
(3) $Q[\{x.\mathsf{Pf}_1 = y.\mathsf{Pf}_2\}] \overset{\mathcal{T}}{=} Q[x.\mathsf{Pf}_1 = y.\mathsf{Pf}_2]$
(4) $Q[\{Q_1 \wedge Q_2\}] \overset{\mathcal{T}}{=} Q[\{Q_1\} \wedge \{Q_2\}]$
(5) $Q[\{Q_1 \vee Q_2\}] \overset{\mathcal{T}}{=} Q[\{Q_1\} \vee \{Q_2\}]$ iff (i)
(6) $Q[\{\exists x.Q_1\}] \overset{\mathcal{T}}{=} Q[\exists x.\{Q_1\}]$ iff (ii)

where

(i) $\mathcal{T} \models C_{Q[\cdot]} \sqcap E_{Q_1} \sqcap E_{Q_2} \sqsubseteq \bot$
(ii) $\mathcal{T} \models C_{Q[\cdot]} \sqcap E_{Q_1} \sqsubseteq$
$C_{Q[\cdot]} \sqcap E_{Q_1} : \mathsf{Fv}(\exists x.Q_1) \cup \mathsf{Fv}(Q[\cdot]) \rightarrow x.$

Proof *(1) is immediate from the definition of $\mathcal{QL}$ semantics since nested duplicate elimination operators only determine the* number *of duplicates of a particular answer, but do not affect the existence of the answer itself. (2) and (3) are also immediate since we only consider duplicate-free atoms. (4) follows from the* multiplicative *nature of conjunction (and the fact that $1 = 1 \cdot 1$). Finally, (5) and (6) follow from Theorems 3 and 2, observing that the context $Q[\cdot]$ behaves as a conjunction with other subqueries.* □

Example 6 Now consider both $\mathcal{QL}$ queries in Example 3. Applying the above rewrite rules would obtain the following equivalent formulations:

$$\begin{aligned}&\exists s.\exists t.\exists c.(\mathtt{STUDENT}(s) \wedge \mathtt{TAKES}(t) \wedge \mathtt{CLASS}(c)\\&\quad \wedge\ s.name = n.id\\&\quad \wedge\ s.id = t.student \wedge c.time = t.class.time\\&\quad \wedge\ c.inst.dept.name = p1 \wedge c.num = p2), \text{ and}\end{aligned}$$

$$\begin{aligned}&(\exists s.(\mathtt{STUDENT}(s) \wedge s.name = n.id)\\&\quad \vee (\exists d.(\mathtt{DEPT}(d) \wedge d.name = n.id).\end{aligned}$$

Observe that all occurrences of $\mathcal{QL}$'s TDE operators have been removed, and also that straightforward translations of the second version of the queries in Examples 1 and 2 in our introductory comments would obtain them. □

5 Negation and Infinite Interpretations

In this section, we discuss various extensions of our results for positive $\mathcal{QL}$ queries (for which the reasoning is complete).

5.1 Negation

The only remaining query construct of $\mathcal{QL}$ is negation. Here, however, query emptiness becomes undecidable [1], even in the absence of a database schema. We can still approximate the interactions with duplicate elimination as follows:

Theorem 5 *Let $\mathcal{T}$ be TBox that captures a database schema and Q a $\mathcal{QL}$ query. Then the following rewrite rules are correct:*

(7) $\{\neg Q\} \overset{\mathcal{T}}{=} \neg Q$
(8) $\neg\{Q\} \overset{\mathcal{T}}{=} \neg Q$

Proof *Immediate from the definition of negation in $\mathcal{QL}$.* □

To abstract queries with negations, we simply ignore the negated part in the definitions of the concept abstraction of both queries and contexts:

$$E_{\neg Q} = C_{\neg[\cdot]} = \top \text{ and } \mathsf{Fv}(\neg[\cdot]) = \emptyset.$$

With this arrangement, all results about the positive fragment of $\mathcal{QL}$ are preserved.

5.2 Incompleteness for Infinite Duplications

In Sections 3 and 4, we restrict the applicability of the rewrite rules to cases in which the interpretations (i.e., database instances) over which the queries are evaluated are finite. A closer inspection of the proofs of Theorems 2 and 3 reveals the following:

1. The rewrite rules are *sound* for all interpretations; and
2. Completeness is only hindered by the use of *cardinal arithmetic* when dealing with infinite numbers of duplicates of *the same tuple*.

For example, consider the following pair of queries:

$$Q(x) \wedge \{B(x) \vee B(x)\}, \text{ and}$$
$$Q(x) \wedge (\{B(x)\} \vee \{B(x)\}).$$

Observe that the queries do not satisfy the condition in Theorem 3, but that, when $\{x \mapsto 1\} : \aleph_0 \in [\![Q]\!](\mathcal{I})$ and when $\{x \mapsto 1\} : n \in [\![B]\!](\mathcal{I})$ (for arbitrary n), we get

$$\{x \mapsto 1\} : \aleph_0$$

to be the answer to both of the above queries since cardinal arithmetic *obscures* the difference between $\aleph_0$ and $2 \cdot \aleph_0$. An example of a subquery Q with the above answer could be $\exists y.\{B(x)\} \wedge A(y)$, where A is a concept for which the database schema enforces an infinite interpretation (e.g., via functionality and disjointness assertions). A similar example can be shown for the rule in Theorem 2.

Note that infinite interpretations *per se* are not an issue here. Indeed, there are two ways of addressing these issues that we believe merit future investigation:

1. Restrictions to $\mathcal{QL}$ that prevent creating infinitely many duplicates of a single tuple in answers to queries; and
2. Refinement of arithmetic that accounts for infinite numbers of duplicates along the lines of ordinal arithmetic.

6 Conclusions

We have introduced $\mathcal{QL}$, a bag-based query language operating over standard interpretations of feature-based DLs, and have also introduced query rewrite rules to manipulate and ultimately eliminate $\mathcal{QL}$'s TDE operator, an operator that performs expensive *tuple duplicate elimination* on query results. The rules are preconditioned by an object relational schema captured as a DL TBox, and are sound and complete for finite instances of the schema.

In contrast to the approaches commonly used in database theory that are base on query subsumption, we have used a direct approach since questions relating to subsumption of conjunctive queries under duplicate semantic are still open [3] and are known not to be decidable for UCQs [5].

Our approach is also directly applicable in the OBDA setting as an additional final step in query reformulation to SQL backends in which *distinct keyword elimination* and *replacing union with union all* can constitute performance critical rewrites [8].

Future work relates to enhancing the $\mathcal{QL}$ semantics to a more fine-grained appreciation of infinite duplication of answers in order to obtain completeness of the rewriting for infinite instances, as well as extending the technique to other query constructs, such as aggregation [4].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web **2**(1), 43–53 (2011). DOI 10.3233/SW-2011-0029. URL https://doi.org/10.3233/SW-2011-0029
3. Chaudhuri, S., Vardi, M.Y.: Optimization of *real* conjunctive queries. In: ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pp. 59–70 (1993)
4. DeHaan, D., Toman, D., Weddell, G.: Rewriting Aggregate Queries using Description Logics. In: Description Logics 2003, pp. 103–112. CEUR-WS vol.81 (2003)
5. Ioannidis, Y.E., Ramakrishnan, R.: Containment of conjunctive queries: Beyond relations as sets. ACM Trans. Database Syst. **19**(3), 288–324 (1995)
6. Khizder, V., Toman, D., Weddell, G.E.: Adding ABoxes to a Description Logic with Uniqueness Constraints via Path Agreements. In: Proc. International Workshop on Description Logics DL2007, pp. 339–346 (2007)
7. Khizder, V.L., Toman, D., Weddell, G.: Reasoning about Duplicate Elimination with Description Logic. In: DOOD, pp. 1017–1032 (2000)
8. McIntyre, S., Borgida, A., Toman, D., Weddell, G.E.: On limited conjunctions and partial features in parameter-tractable feature logics. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019., pp. 2995–3002 (2019)
9. McIntyre, S., Toman, D., Weddell, G.E.: FunDL - A family of feature-based description logics, with applications in querying structured data sources. In: Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, pp. 404–430 (2019)
10. Pinto, F.D., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013, pp. 561–572 (2013)
11. St. Jacques, J., Toman, D., Weddell, G.E.: Object-relational queries over $\mathcal{CFDI}_{nc}$ knowledge bases: OBDA for the SQL-Literate. In: Proc. IJCAI, pp. 1258–1264 (2016)