First-Order Ontology Mediated Database Querying via Query Reformulation

Diego Calvanese and Enrico Franconi

Abstract We address the problem of query answering with ontologies over databases. We consider first-order ontology systems playing the role of a conceptual model of a database represented as a classical finite relational store, either with an open world or a closed world reading. Queries over the conceptual signature are reformulated into queries over the database signature, so to get the same answer directly via SQL relational database technology. We consider two distinct approaches to reformulation, perfect and exact reformulation. We discuss advantages and disadvantages of each of the two approaches, and we report on some significant results appeared in the literature.

1 Introduction

We address the problem of query answering with ontologies over databases. An ontology provides a conceptual view of the database and consists of constraints on a vocabulary *extending* the basic vocabulary of the data. Querying a database using the terms in such a richer ontology allows for more flexibility than directly using only the basic vocabulary of the relational database. In the proposed framework, we analyse the case when a query expressed in ontology terms can be reformulated as a query expressed directly in database terms, so that it can be evaluated using standard SQL relational technology, which is very efficient in the size of the data.

We consider first-order ontology systems (that is, expressible as first-order theories) playing the role of a conceptual model of a database represented as a classical finite relational store, called the *locally-closed world* [13, 14], *exact views* [15, 25, 26], or, in our case, a *DBox* [19]. A DBox is a set of ground atoms that semantically behaves like a database, i.e., the interpretation of the database predicates in

D. Calvanese · E. Franconi (✉)
KRDB Research Centre, Free University of Bozen-Bolzano, Bolzano, Italy
e-mail: franconi@inf.unibz.it

D. Calvanese
e-mail: calvanese@inf.unibz.it
URL: http://www.inf.unibz.it/krdb/

S. Flesca et al. (eds.), *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, Studies in Big Data 31, DOI 10.1007/978-3-319-61893-7_10

the DBox is exactly equal to the database relations. We say that the DBox predicates are *closed*, i.e., their extensions are the same across all the interpretations. On the other hand, we may have *incomplete* data that behaves as ground atoms in classical first-order logic, or ABoxes in classical description logics: in this case, predicates holding this data are *open*, i.e., their interpretation may be extended in different ways across different interpretations.

As an example, consider an open ABox $\mathcal{A} = \{\mathsf{Person}(\mathsf{john})\}$ and alternatively a closed DBox with the same data $\mathcal{D} = \{\mathsf{Person}(\mathsf{john})\}$. In every model $\mathcal{I}$ of the ABox $\mathcal{A}$ the extension of the predicate Person includes john: namely, $\mathsf{Person}^{\mathcal{I}} \supseteq \{\mathsf{john}\}$; instead, in every model $\mathcal{I}$ of the DBox $\mathcal{D}$ the extension of Person is exactly john: namely $\mathsf{Person}^{\mathcal{I}} = \{\mathsf{john}\}$.

The difference between data held as a DBox and data held as an ABox becomes evident when querying. As an example, consider a Boolean negative query $\neg\mathsf{Person}(\mathsf{mary})$ over a given standard relational database, expressed by the DBox $\mathcal{D} = \{\mathsf{Person}(\mathsf{john})\}$. The answer of the query to the DBox is true, because the only specified person is john. On the other hand, if we consider the ABox $\mathcal{A} = \{\mathsf{Person}(\mathsf{john})\}$ and evaluate the query over it, the answer is false because the ABox specifies only the necessary facts but not all of them, and, hence, mary still may be a person in some model.

The *certain answer* to an open query consists of the substitutions that make the query true in *all* the models of the ontology with the data (ABox or DBox): so, if a substitution makes the query true only in some models but not in others (namely, it would be only a *possible* answer), then it is not part of the certain answer. Notice that it may indeed be the case that the answer to the query is not necessarily the same among all the models of the ontology with the ABox or DBox. In this case, the query is not fully *determined* by the given source data; indeed, given the database, there is some answer that is possible, but not certain. For expressive ontologies and queries, the possibility of indeterminacy of queries with respect to the data, brings about an increase of the worst-case computational complexity of computing certain answers. Moreover, it has been shown that computing arbitrary certain answers with DBoxes may be strictly harder in data complexity than computing certain answers with ABoxes [10, 17, 28]. Alternatively, to gain in efficiency, we could focus only on queries having the same answer over all the models of the ontology with the data, namely, when the information requested by the query is fully available from the source data without ambiguity. In this way, the indeterminacy disappears, and the complexity of query answering may be lower.

For example, consider an ontology stating that the class Person is partitioned into the classes Male and Female, and that the class Person is a subclass of the class Animal, and assume that there is data only for Person and Male (and that such data is closed, i.e., complete). The query asking for all the animals (for which we do not have directly data) is not fully determined by the data, since there may be animals who are not persons. On the other hand, the query asking for all the female persons is fully determined by the data, since the female persons are exactly all the persons who are not male, which we know exactly given the data.

In this chapter we are interested in answering ontology mediated queries via *query reformulation*: namely the original query is reformulated as a query expressed only in database terms, so that it can be efficiently evaluated directly over the database using standard SQL relational technology.

The mainstream research on query reformulation [21] is based on perfect rewritings with ABoxes and relatively inexpressive ontologies and queries (see, e.g., the *DL-Lite* approach in [2, 8]). Given an ontology and an arbitrary query, its *perfect reformulation* is a formula that, when evaluated over an arbitrary database, is guaranteed to return the certain answers of the original query with respect to the ontology and the same database. In the example above, the perfect reformulation of the query asking for all the animals is the query asking for all the persons, since those are the individuals which for sure are in the answer of the original query. On the other hand, we could construct a stronger reformulated query expressed in terms of database predicates, by requiring it to be also logically equivalent to the original query with respect to the ontology. This condition still obviously guarantees to give the same certain answer to an arbitrary database with respect to the ontology as the original query. In addition, these equivalent reformulations (called *exact reformulations*) exist if and only if the answer of the original query itself is fully determined by the given source data [4, 19]. So, in the case of exact reformulations we can deal with more expressive ontologies and queries, at the cost of being able to answer only queries determined by the data. In our example, there is no exact reformulation for the query asking for all the animals, but there is an exact reformulation for the query asking for all the females. It should be noticed that every exact reformulation is perfect, but not vice-versa.

In this chapter we survey and compare within a common formal framework the two approaches to ontology mediated query answering: via perfect reformulations in Sect. 3, and via exact reformulations in Sect. 4. We discuss advantages and disadvantages of each of the two approaches, and we report on the most relevant results appeared in the literature.

2 First-Order Ontologies and Databases

Let $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ be a classical function-free first-order language with equality over a signature $(\mathbb{C}, \mathbb{P})$, where $\mathbb{C}$ is a countably infinite set of *constants*, and $\mathbb{P}$ is a countably infinite set of *predicates* with associated arities. We consider $\mathbb{P}$ partitioned into a set $\mathbb{P}_D$ of *database predicates*, which intuitively we consider closed, and a set $\mathbb{P}_T$ of so-called *TBox predicates*, which intuitively we consider open. In the rest of the paper we will use $\mathcal{L}$ to denote some chosen fragments of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$.

We denote with $\mathbb{P}_{\{\varphi_1,\ldots,\varphi_n\}}$ the set of all predicates and with $\mathbb{C}_{\{\varphi_1,\ldots,\varphi_n\}}$ the set of all constants occurring in the formulas $\varphi_1, \ldots, \varphi_n$. For simplicity, we write $\mathbb{P}_\varphi$ (resp., $\mathbb{C}_\varphi$) instead of $\mathbb{P}_{\{\varphi\}}$ (resp., $\mathbb{C}_{\{\varphi\}}$). When we want to make the free variables X of a (possibly open) formula φ explicit, we write the formula as $\varphi_{[X]}$.

A (possibly empty) *finite* set of closed formulas in $\mathcal{L}$ is called an *ontology* (or knowledge base). We consider the following specific components of an ontology $\mathcal{K}$:

- The *database* (or *DBox*) $\mathcal{D}$ is the set of ground atoms in $\mathcal{K}$ of the form $P(c_1, \ldots, c_n)$, where P is an n-ary predicate in $\mathbb{P}_D$, and $c_i \in \mathbb{C}$, for $i \in \{1, \ldots, n\}$.
- The *ABox* $\mathcal{A}$ of $\mathcal{K}$ is defined analogously to the DBox, except that its predicates are in $\mathbb{P}_T$ (instead of $\mathbb{P}_D$). Clearly, the sets of ABox and DBox predicates are disjoint.
- The *TBox* $\mathcal{T}$ is the set of formulas in $\mathcal{K}$ over predicates in $\mathbb{P}_T$ only, but that are not part of the ABox of $\mathcal{K}$.
- The (sound GAV) *mapping* $\mathcal{M}$ is the set of formulas in $\mathcal{K}$ of the form $\forall X.\varphi_{[X]} \rightarrow P(X)$, where $\mathbb{P}_\varphi \subseteq \mathbb{P}_D$ and $P \in \mathbb{P}_T$.
- The *views* $\mathcal{V}$ is the set of formulas in $\mathcal{K}$ of the form $\forall X.\varphi_{[X]} \leftrightarrow P(X)$, where $\mathbb{P}_\varphi \subseteq \mathbb{P}_D$ and $P \in \mathbb{P}_T$.

Notice that, in general, an ontology might contain additional formulas with respect to those in these specific five components. For a DBox, ABox, TBox, mapping, views, or ontology $\mathcal{B}$, we denote with $\mathbb{P}_\mathcal{B}$ the set of all predicates, and with $\mathbb{C}_\mathcal{B}$ the set of all constants appearing in $\mathcal{B}$.

As usual, an *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ consists of a non-empty set, the *domain* $\Delta^\mathcal{I}$, and an interpretation function $\cdot^\mathcal{I}$ defined over constants and predicates of the signature, such that for every pair of database constants $c_1, c_2 \in \mathbb{C}$, if $c_1 \neq c_2$ then $c_1^\mathcal{I} \neq c_2^\mathcal{I}$ (i.e., *we assume unique names* (UNA)). An interpretation $\mathcal{I}$ *embeds a database* $\mathcal{D}$, if it holds *(i)* $c^\mathcal{I} = c$ for every database constant $c \in \mathbb{C}_\mathcal{D}$ (i.e., we make the *standard name assumption* (SNA) on the database constants), and that *(ii)* $(c_1, \ldots, c_n) \in P^\mathcal{I}$ *if and only if* $P(c_1, \ldots, c_n) \in \mathcal{D}$. We use $\mathcal{E}(\mathcal{D})$ to denote the set of all interpretations embedding a database $\mathcal{D}$. In other words, in every interpretation embedding a database $\mathcal{D}$, the interpretation of every database predicate is always the same, and it is given exactly by its content in the database; this is, in general, not the case for the interpretation of the TBox (i.e., non-database) predicates. We say that the database predicates are *closed*, while the other predicates, i.e., the TBox predicates, are *open* and may be interpreted differently in different interpretations. In an open world, an interpretation $\mathcal{I}$ *soundly embeds* a database $\mathcal{D}$ if it holds that $(c_1, \ldots, c_n) \in P^\mathcal{I}$ if (but *not* only if) $P(c_1, \ldots, c_n) \in \mathcal{D}$.

As usual, an interpretation in which a closed formula is true according to the classical FOL definitions is called a *model* of the formula; the set of all models of a formula φ (resp., ontology $\mathcal{K}$) is denoted as $Mod(\varphi)$ (resp., $Mod(\mathcal{K})$). A database $\mathcal{D}$ is *legal for an ontology* $\mathcal{K}$ if there exists a model of $\mathcal{K}$ embedding $\mathcal{D}$. In the following, we will consider only consistent non-tautological ontologies and legal databases.

Given an interpretation $\mathcal{I}$ of a language $\mathcal{L}(\mathbb{P}, \mathbb{C})$, and $\mathbb{P}' \subseteq \mathbb{P}$ and $\mathbb{C}' \subseteq \mathbb{C}$, we denote with $\mathcal{I}|_{(\mathbb{P}', \mathbb{C}')}$ the interpretation identical to $\mathcal{I}$, except that the interpretation function $\cdot^{\mathcal{I}|_{(\mathbb{P}', \mathbb{C}')}}$ is defined only for the constants and predicates of the smaller signature $(\mathbb{P}', \mathbb{C}')$.

Let X be a set of variable symbols and $\mathbb{S}$ a set. A *substitution* is a total function $\Theta : X \rightarrow \mathbb{S}$ assigning an element in $\mathbb{S}$ to each variable in X, including the empty substitution ε when $X = \emptyset$. Domain and image (or range) of a substitution Θ are written as $dom(\Theta)$ and $rng(\Theta)$ respectively. Given a formula $\varphi_{[X]}$, an interpretation

$\mathcal{I}$, and a substitution $\Theta : X \to \mathbb{C}$, we use $\mathcal{I} \models \varphi_{[X/\Theta]}$ to denote that $\varphi_{[X]}$ is true in $\mathcal{I}$ with its free variables substituted according to $\Theta : X \to \mathbb{C}$. Given a formula $\varphi_{[X]}$, an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, and a substitution $\Theta : X \to \Delta^{\mathcal{I}}$ (i.e., a variable assignment), we use $\mathcal{I}, \Theta \models \varphi$ to denote that $\varphi_{[X]}$ is true in $\mathcal{I}$ with its free variables interpreted according to $\Theta : X \to \Delta^{\mathcal{I}}$.

Queries. A *query* is a (possibly closed) formula. The number of free variables of the query is called its *arity*. When the arity is 0 (i.e., the query is a closed formula), we call the query *Boolean*. The *(certain) answer* $\mathsf{cert}(Q_{[X]}, \mathcal{K})$ of a query $Q_{[X]}$ over an ontology $\mathcal{K}$ containing a database $\mathcal{D}_{\mathcal{K}}$ is the set of substitutions with constants that make the query true for all models of $\mathcal{K}$ that embed its database $\mathcal{D}_{\mathcal{K}}$. Formally:

$$\mathsf{cert}(Q_{[X]}, \mathcal{K}) \;=\; \{\Theta : X \to \mathbb{C} \;\mid\; \text{for all } \mathcal{I} \in \mathit{Mod}(\mathcal{K}) \cap \mathcal{E}(\mathcal{D}_{\mathcal{K}}) : \mathcal{I} \models Q_{[X/\Theta]}\}$$

For a Boolean query Q, $\mathsf{cert}(Q, \mathcal{K})$ is either the empty set (corresponding to false), or the empty assignment (corresponding to true).

In this chapter we are interested in the *query answering problem*: given an ontology $\mathcal{K}$ and a query Q, compute $\mathsf{cert}(Q_{[X]}, \mathcal{K})$. To study its computational complexity, we consider the associated decision problem (sometimes called the *recognition problem for query answering*): given an ontology $\mathcal{K}$, a query Q, and a substitution Θ, decide whether $\Theta \in \mathsf{cert}(Q_{[X]}, \mathcal{K})$. The *combined complexity* of the problem is the complexity measured in the combined size of the ontology and the query, while the *data complexity* [38] is the complexity measured only in the size of the data, i.e., the ABox and the DBox/database.

It has been shown that it is possible to weaken the standard name assumption for the database constants without changing the certain answers, by just assuming *unique names* [19]. So, the proposed framework is entirely classical, namely it can be completely represented in classical first-order logic with equality.

In order to capture exactly the expressivity of Relational Algebra and of core query languages used in databases, notably SQL [1], we define now *domain independent* formulas with respect to a given ontology, by adapting the classical database definition of domain independence to our ontology based framework.

A formula $Q_{[X]}$ is *domain independent with respect to an ontology* $\mathcal{K}$ if and only if for every two models $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ of $\mathcal{K}$ that agree on the interpretation of the predicates and constants (i.e., $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}}$), and for every substitution $\Theta : X \to \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$ we have:

$$\mathit{rng}(\Theta) \subseteq \Delta^{\mathcal{I}} \text{ and } \mathcal{I}, \Theta \models Q_{[X]} \quad \text{if and only if} \quad \mathit{rng}(\Theta) \subseteq \Delta^{\mathcal{J}} \text{ and } \mathcal{J}, \Theta \models Q_{[X]}.$$

The above definition reduces to the database definition of domain independence whenever the ontology is empty. The problem of checking whether a first-order logic formula is domain independent is undecidable [1, 11]. The well known *safe-range* syntactic fragment of first-order logic introduced by Codd is an *equally expressive* language; indeed any safe-range formula is domain independent, and any domain independent formula can be easily transformed into a logically equivalent safe-range

formula. This transformation implements the idea that the range of every variable of domain independent formula should be restricted by some "guard" bounding its extension.

An *SQL query* is a domain independent first order formula [1]. A *conjunctive query* (CQ) is an SQL query constructed using conjunction and existential quantification only. A *union of conjunctive queries* (UCQ) is a disjunction of CQs, all of the same arity. While in UCQs disjunction can only be used as the outermost operator, *positive queries* (PQs) allow for the arbitrary use of conjunction, disjunction, and existential quantification (but they rule out negation). In fact, PQs, which are the positive fragment of domain independent first-order queries, have the same expressive power as UCQs, although they may be exponentially more succinct.

Description logics ontologies. Description logics are relevant fragments of first-order logic with good computational properties. We consider here the expressive description logic $\mathcal{ALCHOIQ}$ as a fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$. In $\mathcal{ALCHOIQ}$, only unary and binary predicates are allowed, called respectively *concepts* and *roles*, together with constants and the equality predicate. $\mathcal{ALCHOIQ}$ formulas φ (also called *DL axioms*), concepts C, and roles R are defined according to the following syntax:

$$\begin{array}{rcl} \varphi & \Longrightarrow & C \sqsubseteq C' \mid R \sqsubseteq R' \mid C(c) \mid R(c_1, c_2) \\ C, C' & \Longrightarrow & A \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \mid \{c_1, \ldots, c_n\} \mid \\ & & \exists R.C \mid \forall R.C \mid \geq k\, R.C \mid \leq k\, R.C \\ R, R' & \Longrightarrow & S \mid S^- \end{array}$$

where A denotes a concept name (a unary TBox predicate), S a role name (a binary TBox predicate), c, $c_1, \ldots, c_n$ constants (also called *individuals*), and k a positive integer. Other constructs in the language are defined as shortcuts: $\top \equiv A \sqcup \neg A$ for some A; $\bot \equiv \neg\top$; $\exists R \equiv \exists R.\top$; $\geq n\, R \equiv \geq n\, R.\top$; $\leq n\, R \equiv \leq n\, R.\top$. A *DL ontology* is a set of DL axioms, where the DL axioms of the form $C \sqsubseteq C'$ and $R \sqsubseteq R'$ constitute the *DL TBox*, and the DL axioms of the form $C(c)$ and $R(c_1, c_2)$ constitute the *DL ABox*.

We introduce also some significant sublanguages of $\mathcal{ALCHOIQ}$. The description logic $\mathcal{ALCHOI}$ is the fragment of $\mathcal{ALCHOIQ}$ without the cardinality operators $\geq k\, R.C$ and $\leq k\, R.C$, while $\mathcal{ALCHOQ}$ is the fragment of $\mathcal{ALCHOIQ}$ without the inverse operator for roles P^-.

The lightweight logics of the *DL-Lite* family [2, 8] are description logics specifically tuned for accessing large amounts of data. Here we adopt the simple variant *DL-Lite*$_\mathcal{R}$ presented in [8], but all our considerations can be extended also to some of the more expressive variants of *DL-Lite* that are studied in [2] (specifically, those for which query answering is first-order rewritable, see Sect. 3). We use simply *DL-Lite* when our statements apply in general to such logics. In *DL-Lite*$_\mathcal{R}$, DL axioms, concepts, and roles are defined according to the following syntax:

Fig. 1 Mapping $\cdot^\dagger$ from an $\mathcal{ALCHOIQ}$ ontology to a first-order ontology

$$
\begin{aligned}
(C_1 \sqsubseteq C_2)^\dagger &= \forall x.\, {C_1}^\dagger(x) \rightarrow {C_2}^\dagger(x) \\
(R_1 \sqsubseteq R_2)^\dagger &= \forall x, y.\, {R_1}^\dagger(x, y) \rightarrow {R_2}^\dagger(x, y) \\
(C(c))^\dagger &= C^\dagger(c) \\
(R(c_1, c_2))^\dagger &= R^\dagger(c_1, c_2) \\
A^\dagger &= \lambda x.\, A(x) \\
S^\dagger &= \lambda x, y.\, S(x, y) \\
(\neg C)^\dagger &= \lambda x.\, \neg C^\dagger(x) \\
(C_1 \sqcap C_2)^\dagger &= \lambda x.\, {C_1}^\dagger(x) \wedge {C_2}^\dagger(x) \\
(C_1 \sqcup C_2)^\dagger &= \lambda x.\, {C_1}^\dagger(x) \vee {C_2}^\dagger(x) \\
(\exists R.C)^\dagger &= \lambda x.\, \exists y. R^\dagger(x, y) \wedge C^\dagger(y) \\
(\forall R.C)^\dagger &= \lambda x.\, \forall y. R^\dagger(x, y) \rightarrow C^\dagger(y) \\
(\geq k\, R.C)^\dagger &= \lambda x.\, \exists^{\geq k} y.\, R^\dagger(x, y) \wedge C^\dagger(y) \\
(\leq k\, R.C)^\dagger &= \lambda x.\, \exists^{\leq k} y.\, R^\dagger(x, y) \wedge C^\dagger(y) \\
\{c_1, \ldots, c_n\}^\dagger &= \lambda x.\, x = c_1 \vee \cdots \vee x = c_n \\
(S^-)^\dagger &= \lambda x, y.\, S(y, x)
\end{aligned}
$$

$$
\begin{aligned}
\varphi &\Longrightarrow B \sqsubseteq B' \mid B \sqsubseteq \neg B' \mid R \sqsubseteq R' \mid R \sqsubseteq \neg R' \mid A(c) \mid S(c_1, c_2) \\
B, B' &\Longrightarrow A \mid \exists R \\
R, R' &\Longrightarrow S \mid S^-
\end{aligned}
$$

Here, B, B' are called *basic concepts*, and they denote either an atomic concept A, or the projection of a role S on its first component ($\exists S$) or on its second component ($\exists S^-$). We observe that in TBox axioms, negation is used only on the right-hand side of inclusions, i.e., to express disjointness between concepts and between roles. Moreover, *DL-Lite*$_\mathcal{R}$ restricts ABox axioms to use concept and role names only, as opposed to complex concept expressions allowed in $\mathcal{ALCHOIQ}$.

We observe that, despite its simplicity, *DL-Lite*$_\mathcal{R}$ is able to capture the essential features of most conceptual modeling formalisms, such as UML Class Diagrams or Entity-Relationship schemata (see, e.g., [7]).

Description logics versus first-order ontologies. A DL ontology can be translated into first-order logic formulas by means of a mapping function $\cdot^\dagger$. In this translation, concept and role names correspond respectively to unary and binary predicate symbols in $\mathbb{P}_T$, and individuals corresponds to constant symbols in $\mathbb{C}$. Specifically, the mapping $\cdot^\dagger$ from an $\mathcal{ALCHOIQ}$ ontology to a first-order ontology is defined in Fig. 1. In our translation, we have made use of first-order logic with counting quantifiers [31], but such quantifiers are just syntactic sugar that can be easily translated away with just a linear blowup in the size of the formula (assuming numbers are represented in unary).

Notice that, since a DL ontology is constructed over TBox (i.e., open) predicates of $\mathbb{P}_T$ only, the translation of a DL ontology to first-order, results in the TBox and ABox part of a first-order ontology. These are obtained respectively from the translation of the DL TBox and the DL ABox. We may then combine such a first-order ontology with additional axioms that also make use of database predicates in $\mathbb{P}_D$, e.g., a

database (or DBox) part consisting of ground atoms over $\mathbb{P}_D$, a sound GAV mapping part, a view part, or additional more general formulas.

3 Perfect Query Reformulations

We illustrate now the approach to query reformulation based on *perfect rewriting*, which was first introduced for answering UCQs over *DL-Lite* ontologies through the *PerfectRef* algorithm [8], and then extended to several other DLs [23, 29, 33], including also more expressive members of the *DL-Lite* family [2]. Such DLs share with *DL-Lite* some crucial properties that are necessary to make a rewriting based approach efficient. We illustrate first the approach for the case of UCQs over standard *DL-Lite*$_\mathcal{R}$ ontologies, constituted by a TBox and an ABox only. We come back later to how to extend the approach to take into account the presence of a database and of sound GAV mappings.

We recall that we are interested only in the case where the ontology is consistent. Indeed, the case where the ontology is inconsistent is not really of interest, since then the certain answers of every query is the set of all possible variable substitutions. However, we address afterwards also the problem of checking consistency.

The key idea at the basis of the rewriting approach is to strictly separate the processing done with respect to the TBox (providing the intensional level of the ontology) from the processing done by taking into account the ABox (i.e., the extensional level). More precisely, let $\mathcal{K}$ be a *DL-Lite*$_\mathcal{R}$ ontology with TBox $\mathcal{T}$ and ABox $\mathcal{A}$, and let Q be a UCQ over $\mathcal{K}$.

1. The query Q is first processed and rewritten into a new query Q', by compiling into Q' the *positive inclusion assertions* of the TBox $\mathcal{T}$, i.e., those inclusion assertions that contain no negation in the right-hand side.
2. The rewritten query Q' is then evaluated over the ABox $\mathcal{A}$, as if $\mathcal{A}$ was a (closed) database (and not considering further the TBox).

In this way, computing the certain answers is essentially reduced to query evaluation over a database instance. Since Q' does not depend on the ABox, the data complexity of the whole query answering algorithm is the same as the data complexity of evaluating Q' over the ABox. A crucial property for *DL-Lite* is that, in the case where Q is a UCQ, the query Q' is also a UCQ. Hence, the data complexity of the whole query answering algorithm is in AC^0, which is the complexity of evaluating a UCQ (or, more in general, a first-order query) over a relational database.

Canonical model. The rewriting based approach relies in an essential way on the *canonical model property*, which holds for *DL-Lite* and for the horn variants of many other DLs [12, 24]. Such property ensures that every satisfiable ontology $\mathcal{K}$ admits a canonical model $\mathcal{I}_c$ that is the least constrained model among all models of $\mathcal{K}$, and that can be homomorphically embedded in all other models. This in turn implies that the canonical model correctly represents *all* the models of $\mathcal{K}$ with respect to the

Fig. 2 Rewriting of query atoms in *DL-Lite*$_{\mathcal{R}}$

$$
\begin{array}{rlcl}
A_1 \sqsubseteq A_2 & \ldots, A_2(x), \ldots & \rightsquigarrow & \ldots, A_1(x), \ldots \\
\exists S \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, S(x, \_), \ldots \\
\exists S^- \sqsubseteq A & \ldots, A(x), \ldots & \rightsquigarrow & \ldots, S(\_, x), \ldots \\
A \sqsubseteq \exists S & \ldots, S(x, \_), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
A \sqsubseteq \exists S^- & \ldots, S(\_, x), \ldots & \rightsquigarrow & \ldots, A(x), \ldots \\
\exists S_1 \sqsubseteq \exists S_2 & \ldots, S_2(x, \_), \ldots & \rightsquigarrow & \ldots, S_1(x, \_), \ldots \\
S_1 \sqsubseteq S_2 & \ldots, S_2(x, y), \ldots & \rightsquigarrow & \ldots, S_1(x, y), \ldots \\
S_1 \sqsubseteq S_2^- & \ldots, S_2(x, y), \ldots & \rightsquigarrow & \ldots, S_1(y, x), \ldots \\
\ldots & & &
\end{array}
$$

problem of answering positive queries (and in particular, UCQs). In other words, for every UCQ Q, we have that $\mathsf{cert}(Q, \mathcal{K})$ is contained in the result of the evaluation of Q over the canonical model.[1] Intuitively, the canonical model $\mathcal{I}_c$ for a *DL-Lite* ontology $\mathcal{K}$ contains the ABox $\mathcal{A}$ of $\mathcal{K}$, and in addition might contain existentially implied objects, whose existence is enforced by the TBox axioms with $\exists R$ in the right-hand side. For example, if the TBox contains an axiom $\mathsf{Student} \sqsubseteq \exists \mathsf{attends}$, expressing that every student should attend something (presumably a course), and the ABox contains the fact $\mathsf{Student}(\mathsf{john})$, then the canonical model will contain a fact $\mathsf{attends}(\mathsf{john}, o_n)$, where o_n is a newly introduced object.

First-Order rewritability. For simplicity, let us consider the case where we want to answer a Boolean UCQ $Q = \bigvee_i q_i$, where each q_i is a Boolean CQ. In order for some q_i to contribute to the answer to Q, there must exist a homomorphic embedding of all atoms of q_i into facts of the canonical model $\mathcal{I}_c$. We call such embedding a *match* of q_i into $\mathcal{I}_c$. However, $\mathcal{I}_c$ is in general infinite, hence we cannot evaluate q_i over $\mathcal{I}_c$ by effectively computing $\mathcal{I}_c$ and then searching for a match of q_i into $\mathcal{I}_c$. Instead, each CQ q_i is rewritten into a UCQ Q_i in such a way that, whenever q_i has a match in some portion of $\mathcal{I}_c$, then there will be a CQ among those in Q_i that has a corresponding match in the ABox part of $\mathcal{I}_c$. Informally, the rewriting algorithm initializes a set *Rew* of CQs to $\bigcup_i \{q_i\}$ (i.e., to the CQs in the input query Q), and processes each yet unprocessed query q_i in *Rew* by adding to *Rew* also all rewritings of q_i. For each atom α in q_i, it checks whether α can be rewritten by using one of the positive inclusions in the TBox, and if so, adds to *Rew* the CQ obtained from q_i by rewriting α. The rewriting of an atom uses a positive inclusion axiom as rewriting rule, applied from right to left, to compile away the knowledge represented by the positive inclusion itself. For example, using the inclusion $A_1 \sqsubseteq A_2$, an atom of the form $A_2(x)$ is rewritten to $A_1(x)$. Alternatively, we can consider this rewriting step as the application of standard resolution between the query and the inclusion $A_1 \sqsubseteq A_2$, viewed as the (implicitly universally quantified) formula $A_1(x) \rightarrow A_2(x)$. Other significant cases of rewritings of atoms are depicted in Fig. 2, where each inclusion axiom in the left-most column accounts for rewriting the atom to the left

[1] Note that, since the domain of the canonical model contains the individuals of the ontology, the evaluation of a query over such model can indeed return a set of individuals.

of $\rightsquigarrow$ into the atom to the right of it. We have used "_" to denote a variable that occurs only once in the CQ (counting separately occurrences as answer variable of the CQ). Besides rewriting atoms, a further processing step applied to q_i is to consider each pair of atoms α_1, α_2 occurring in the body of q_i that *unify*, and replace them with a single atom, also applying the most general unifier to the whole of q_i. In this way, variables that in q_i occur multiple times, might be replaced by an "_", and hence inclusion assertions might become applicable that were not so before the atom-unification step (cf. the rewriting rules in Fig. 2 *requiring* the presence of "_").

The above presented rewriting technique, realized through *PerfectRef*, allows us to establish that answering UCQs over consistent *DL-Lite*$_\mathcal{R}$ ontologies is *first-order rewritable*, i.e., the problem of computing certain answers over a consistent ontology can be reduced to the problem of evaluating a first-order query over the ABox of the ontology viewed as a database (with complete information). Specifically, let $\mathsf{rew}(Q, \mathcal{T})$ denote the UCQ obtained as the result of applying *PerfectRef* to a UCQ Q and a *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$. Then, for every ABox $\mathcal{A}$ such that the ontology $\mathcal{T} \cup \mathcal{A}$ is consistent, we have that

$$\mathsf{cert}(Q, \mathcal{T} \cup \mathcal{A}) \;=\; \mathsf{eval}(\mathsf{rew}(Q, \mathcal{T}), \mathcal{A})$$

where $\mathsf{eval}(\mathsf{rew}(Q, \mathcal{T}), \mathcal{A})$ denotes the evaluation of the UCQ $\mathsf{rew}(Q, \mathcal{T})$ over the ABox $\mathcal{A}$ viewed as a database (i.e., a first-order interpretation).

Ontology satisfiability. The rewriting of a UCQ Q with respect to a TBox $\mathcal{T}$ computed by *PerfectRef* depends only on the set of positive inclusion axioms in $\mathcal{T}$, while disjointness axioms (i.e., inclusion axioms containing a negated basic concept on the right-hand side) do not play any role in such a process. Indeed, the proof of correctness of *PerfectRef* [8], which is based on the canonical model property of *DL-Lite*$_\mathcal{R}$, shows that these kinds of axioms have to be considered only when verifying the consistency of the ontology. Once consistency is established, they can be ignored in the query rewriting phase. In fact, inconsistency of a *DL-Lite*$_\mathcal{R}$ ontology is due to the presence of disjointness axioms and their interaction with positive inclusion axioms. Such interaction can itself be captured by constructing a Boolean UCQ encoding the violation of disjointness axioms, rewriting such a UCQ with respect to the positive inclusion axioms, and checking whether its evaluation over the ABox returns true. This in turn shows that also the problem of checking consistency of a *DL-Lite*$_\mathcal{R}$ ontology is first-order rewritable [8].

Accessing a database via sound mappings. We have so far considered only the TBox and ABox components of a *DL-Lite* ontology. This is coherent with the implicit assumption that we are dealing with systems in which the data is directly represented, in terms of unary and binary ABox facts, in a form that is perfectly compatible with the intensional layer provided by the TBox. In many real-world scenarios, however, the data is not provided by ABox facts, but by legacy data sources, in (possibly large) relational tables of arbitrary arity. We represent such data sources through the *database* part $\mathcal{D}$ of an ontology. To provide access to the data sources via the intensional level of the ontology, we can rely on the *sound (GAV) mapping* component

$\mathcal{M}$ of an ontology. This setting is known in the literature as *ontology-based data access* (OBDA), and has been investigated extensively in recent years [6, 7, 30].

We recall that, in the case of a DL ontology, the mapping $\mathcal{M}$ consists of axioms of the form:

$$\forall x.\varphi^{A}_{[x]} \rightarrow A(x) \qquad \text{or} \qquad \forall x, y.\varphi^{S}_{[x,y]} \rightarrow S(x, y),$$

where the predicate symbols in $\varphi^{A}_{[x]}$ and $\varphi^{S}_{[x,y]}$ in the left part of the implication are among the database predicates $\mathbb{P}_D$, while A and S in the right part of the implication are respectively a concept and a role name of the TBox. Also, we can assume without loss of generality, that for each concept or role name N we have exactly one such mapping axiom, where $\varphi^{N}_{[X]}$ is the formula in the antecedent of the implication.

The query answering technique based on perfect rewriting can be extended to deal also with sound GAV mappings of this form, so as to rewrite the query into one that contains only database predicates [30]. Let us denote with $\mathcal{M}(\mathcal{D})$ the set of facts obtained by populating each concept or role name N of the TBox with precisely the facts retrieved through the mapping axiom associated to N, by evaluating φ^{N} over $\mathcal{D}$, i.e., with $\mathsf{eval}(\varphi^{N}, \mathcal{D})$. Considering that the mapping formulas are all implications, if such formulas are satisfied in a model $\mathcal{I}$, then they are also satisfied in every model $\mathcal{I}'$ that extends with respect to $\mathcal{I}$ the interpretation of the concept and role names in $\mathbb{P}_T$, i.e., such that $N^{\mathcal{I}} \subseteq N^{\mathcal{I}'}$ for every concept or role name N. It follows that for a positive query Q formulated over the predicates of the TBox $\mathcal{T}$, the certain answers $\mathsf{cert}(Q, \mathcal{T} \cup \mathcal{M} \cup \mathcal{D})$ coincide with the certain answers computed over the TBox $\mathcal{T}$ together with $\mathcal{M}(\mathcal{D})$ considered as an ABox, i.e.,

$$\mathsf{cert}(Q, \mathcal{T} \cup \mathcal{M} \cup \mathcal{D}) = \mathsf{cert}(Q, \mathcal{T} \cup \mathcal{M}(\mathcal{D}))$$

To compute $\mathsf{cert}(Q, \mathcal{T} \cup \mathcal{M}(\mathcal{D}))$, we can in principle follow the approach illustrated previously: compute $\mathcal{M}(\mathcal{D})$ (which is an ordinary finite ABox), and then evaluate over it the perfect reformulation $\mathsf{rew}(Q, \mathcal{T})$. However, this would require a potentially expensive and large materialization of $\mathcal{M}(\mathcal{D})$ from the database $\mathcal{D}$. Instead, we can obtain the same result, avoiding the costly materialization step, by proceeding as follows:

1. Compute the perfect reformulation $Q' = \mathsf{rew}(Q, \mathcal{T})$ of Q with respect to the (positive) inclusion axioms in the TBox part $\mathcal{T}$ of $\mathcal{K}$.
2. Compute the *unfolding* of Q' with respect to $\mathcal{M}$, denoted $\mathsf{unf}(Q', \mathcal{M})$, by replacing each (concept or role) predicate N appearing in Q' with the corresponding formula $\varphi^{N}_{[X]}$ in the left-hand side of the mapping axiom $\forall X.\varphi^{N}_{[X]} \rightarrow N(X)$.
3. Evaluate $\mathsf{unf}(Q', \mathcal{M})$ over the database $\mathcal{D}$.

The correctness of this approach follows from the following result:

Theorem 1 ([8, 30]) *Let $\mathcal{K}$ be a DL-Lite$_{\mathcal{R}}$ ontology consisting of a TBox $\mathcal{T}$, a mapping $\mathcal{M}$ and a database $\mathcal{D}$, and let Q be a UCQ formulated over $\mathcal{T}$. Then:*

$$\mathsf{cert}(Q, \mathcal{T} \cup \mathcal{M} \cup \mathcal{D}) = \mathsf{eval}(\mathsf{unf}(\mathsf{rew}(Q, \mathcal{T}), \mathcal{M}), \mathcal{D})$$

Complexity of query evaluation. Summarizing the above results, and considering that evaluating a first-order query (and hence a UCQ) over a database is in AC^0 in data complexity, one obtains that answering UCQs over *DL-Lite*$_\mathcal{R}$ ontologies has the same data complexity as evaluating UCQs in plain databases. By analyzing the overall rewriting-based query answering technique, and by exploiting a correspondence between the *DL-Lite* family and first-order logic with unary predicates [2], we are able obtain also tight complexity bounds in the size of the TBox (*schema complexity*) and of the overall input (*combined complexity*).

Theorem 2 ([2, 8, 30]) *Answering UCQs over DL-Lite*$_\mathcal{R}$ *ontologies is in* AC^0 *in data complexity,* NLogSpace*-complete in schema complexity, and NP-complete in combined complexity.*

While the above results sound very encouraging from the theoretical point of view, there still remain significant challenges to be addressed to make rewriting based techniques effective also in real world scenarios, where the TBox and/or the data underlying the ABox are very large, and/or queries have a large number of atoms. Indeed, also in the case where one admits rewritings expressed in languages different from UCQs (e.g., arbitrary FOL queries, or non-recursive Datalog), it has recently been shown that the smallest rewritings can grow exponentially with the size of the query [20]. This has led to an intensive and sustained effort aimed at developing techniques to improve query answering over ontologies, such as alternative rewriting techniques [29, 35], techniques combining rewriting with partial materialization of the extensional level [22], and various optimization techniques that take into account also extensional constraints on the underlying data, or in the mapping layer to the relational data sources [32, 34].

4 Exact Query Reformulations

We illustrate in this section the approach to query reformulation with first-order ontologies based on *exact rewritings*, which was first thoroughly analysed in [26] by Nash, Segoufin and Vianu. They addressed the question whether a query can be answered using a set of (exact) views $\mathcal{V}$ by means of an exact rewriting over a database represented as a DBox. The authors defined and investigated the notions of *determinacy* of a query by a set of views and its connection to exact rewriting. Nash, Segoufin and Vianu also studied several combinations of query and view languages trying to understand the expressivity of the language required to express the exact rewriting, and, thus, they obtained results on the *completeness of rewriting languages*. They investigated languages ranging from full first-order logic to conjunctive queries.

The setting considered by Nash, Segoufin and Vianu is weaker than the one studied later by Franconi, Kerhet, and Ngo [18, 19], since in the former setting the ontology

consists only of a set $\mathcal{V}$ of view definitions, while in the latter setting any domain independent first-order ontology with DBox predicates is allowed. The authors also considered the special case of exact reformulations with description logic ontologies and DBoxes [9, 28, 36].

After Franconi, Kerhet, and Ngo, the recent work by Benedikt, ten Cate and Tsamoura [3, 4] analyses the generation of plans to answer queries over DBoxes in the presence of domain independent first-order ontologies, where in addition the DBox relations may have limitations on access patterns to the data (called *access restrictions* or binding patterns). This is the case when one must provide values for one of the attributes of a relation in order to obtain tuples. In this setting, DBox predicates in $\mathbb{P}_D$ as defined in this chapter are fully accessible (as a regular database), while all the other predicates in $\mathbb{P}_T$ are not accessible at all (in fact, they do not provide data).

Toman and Weddell have long advocated the use of exact reformulations for automatic generation of plans that implement user queries under system constraints–a process they called *query compilation* [37]. Their published work focuses on additional extra-logical considerations, such as satisfaction of binding patterns, considerations of inherent ordering of data and the influence of plans on such orderings, and the estimated cost of alternative query plans.

This framework has also been applied to devise the formal foundations of the problems of *view update* and of characterising *unique solutions* in data exchange. In the former problem, a target view of some source database is updatable if the source predicates have an exact reformulation given the view over the target predicates [16]. In the latter problem, unique solutions exist if the target predicates have an exact reformulation given the data exchange mappings over the source predicates [27].

DBoxes versus ABoxes. While the perfect reformulation approach works for data represented in an ABox (or for data in a DBox "connected" to the TBox via a sound mapping), the exact reformulation approach assumes that the data is represented in a DBox. As we have already noticed in the introductory section, ABoxes and DBoxes behave differently in query answering. We observe now that this difference may be relevant for queries from legacy relational database applications.

Indeed, a query mentioning only ABox predicates may have a different answer depending on the presence or absence of an ontology. Consider again the example of a Boolean negative query $\neg\mathsf{Person}(\mathsf{mary})$ over a database expressed by the DBox $\mathcal{D} = \{\mathsf{Person}(\mathsf{john})\}$. As we have said already, the answer of the query to the DBox is true, because the only specified person is john. On the other hand, if we consider the ABox $\mathcal{A} = \{\mathsf{Person}(\mathsf{john})\}$ and evaluate the query over it, the answer is false because the ABox specifies only the necessary facts but not all of them, and, hence, Mary still may be a person in some model. However, if we add the ontology $\{\forall x, y.\, \mathsf{Person}(x) \wedge \mathsf{Person}(y) \rightarrow x = y\}$ that says that there is at most one person, the answer to the query will be now true.

This does not happen by using DBoxes: a query mentioning just DBox predicates can only return an answer that depends only on the DBox predicates, which are

complete. In other words, a DBox preserves the behaviour of legacy application queries over relational databases, also when an ontology is added on top of it.

Domain Independence. As we have seen, domain independence is an important property of a formula that guarantees that the truth value of the formula in an interpretation remains the same regardless of the underlying domain of the interpretation.

An example of a domain independent Boolean query is $\exists x.\,\mathsf{Person}(x)$: if the answer to the query is true for some DBox with a specific domain, it is also true for the same DBox and any other compatible domain. On the other hand, the Boolean query $\forall x.\,\mathsf{Person}(x)$ is not domain independent: if it is true for some DBox with some domain, it is definitely not true for the same DBox but with a larger domain. Consider now the query $Q(x) = \neg\mathsf{Person}(x)$ over the DBox $\{\mathsf{Person}(\mathsf{john})\}$: with domain $\Delta_1 = \{\mathsf{john}, \mathsf{mary}\}$ the query has the answer $\{x \mapsto \mathsf{mary}\}$, while with the extended domain $\Delta_2 = \{\mathsf{john}, \mathsf{mary}, \mathsf{sue}\}$ it has the different answer $\{x \mapsto \mathsf{mary}, x \mapsto \mathsf{sue}\}$. If an infinite domain was considered, the answer would be infinite even with a finite DBox. Indeed, the above query is not domain independent. The query can be "fixed" with a *guard* restricting the range of the free variable, as in $Q'(x) = \mathsf{Animal}(x) \wedge \neg\mathsf{Person}(x)$. For any given database this query returns the same answer with any compatible domain (even an infinite one). $Q'(x)$ is domain independent.

Ontologies are also required to be domain independent. Consider the ontology $\{\exists x.\,\neg\mathsf{Student}(x),\ \exists x.\,\mathsf{Person}(x)\}$, and the DBox $\{\mathsf{Student}(\mathsf{john}), \mathsf{Person}(\mathsf{john})\}$. This DBox with domain $\Delta_1 = \{\mathsf{john}\}$ is inconsistent with respect to the ontology, while it is consistent with respect to the same ontology with domain $\Delta_2 = \{\mathsf{john}, \mathsf{mary}\}$. This happens because the first sentence of the ontology is not domain independent.

Determinacy and exact reformulations. The exact reformulation (also called *explicit definition* by Beth [5]) of a query over some DBox predicates under the ontology is a logically equivalent query under the ontology which uses only DBox predicates. More formally, the *exact reformulation* over the database predicates $\mathbb{P}_D$ under the ontology $\mathcal{K}$ of a query $Q_{[X]}$ is some formula $\widehat{Q}_{[X]}$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$, such that $\mathcal{K} \models \forall X.Q_{[X]} \leftrightarrow \widehat{Q}_{[X]}$ and $\mathbb{P}_{\widehat{Q}} \subseteq \mathbb{P}_D$.

But how can we characterise the existence of an exact reformulation of a query? We need to check that the query is fully determined by the data in the DBox. A query is determined by the data in the DBox if its truth value in every model of the ontology depends *only* on the finite interpretation of the DBox predicates. This notion is formalised as follows. A query $Q_{[X]}$ is *finitely determined* by the database predicates $\mathbb{P}_D$ under the ontology $\mathcal{K}$ *if and only if* for any two models $\mathcal{I}$ and $\mathcal{J}$ of the ontology $\mathcal{K}$, both with a *finite* interpretation to the database predicates $\mathbb{P}_D$, whenever $\mathcal{I}|_{\mathbb{P}_D,\mathbb{C}} = \mathcal{J}|_{\mathbb{P}_D,\mathbb{C}}$ then for every substitution $\Theta : X \to \Delta^{\mathcal{I}}$ we have:

$$\mathcal{I}, \Theta \models Q_{[X]} \quad \textit{if and only if} \quad \mathcal{J}, \Theta \models Q_{[X]}\,.$$

The determinacy of a query with respect to a source database represented in a DBox corresponds to the notion of *implicit definability* of a formula from a set of predicates (the database predicates) as introduced by Beth [5]. The correspondence between

exact reformulations and finite determinacy has been indeed first studied as *projective definability* by Beth himself in 1953 [5].

Intuitively, the answer of a finitely determined query does not depend on the interpretation of non-database predicates. Once the database and a domain are fixed, it is never the case that a substitution would make the query true in some model of the ontology and false in others, since the truth value of a finitely determined query depends only on the interpretation of the database predicates and constants and on the domain (which are fixed). In practice, by focussing on finite determinacy of queries we guarantee that the answers are interpreted as being not only certain, but also *exact*—namely that whatever is not in the answer can never be part of the answer in any possible world.

But how can we characterise the existence of a *domain independent* exact reformulation of a query? This is needed because we want to evaluate the exact reformulation using standard SQL relational technology. The following characterisation theorem answers exactly this question.

Theorem 3 (Semantic characterisation) *Given a set $\mathbb{P}_D$ of database predicates, a domain independent ontology $\mathcal{K}$, and a query $Q_{[X]}$, a domain independent exact reformulation $\widehat{Q}_{[X]}$ of $Q_{[X]}$ over $\mathbb{P}_D$ under $\mathcal{K}$ exists* if and only if *$Q_{[X]}$ is (i) finitely determined by $\mathbb{P}_D$ under $\mathcal{K}$, and (ii) domain independent with respect to $\mathcal{K}$.*

The above theorem shows us the semantic conditions to have an exact domain independent reformulation of a query, but it does not give us a method to compute such reformulation and its equivalent safe-range form. The following theorem gives us sufficient conditions for the existence of an exact safe-range reformulation in any decidable fragment of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ and gives us a constructive way to compute it, if it exists.

Below, we use $\widetilde{Q}$ to denote the formula obtained from a formula Q by uniformly replacing every occurrence of each non-DBox predicate P with a fresh new predicate symbol $\widetilde{P}$. We extend this renaming operator $\widetilde{\cdot}$ to any set of formulas in a natural way. Also, we focus on ontologies and queries in those fragments of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ for which determinacy under models with a finite interpretation of DBox predicates (finite determinacy) and determinacy under models with an unrestricted interpretation of DBox predicates (unrestricted determinacy) coincide. We say that these fragments have *finitely controllable determinacy*.

Theorem 4 (Constructive) *If all the following conditions hold:*

1. $\mathcal{K} \cup \widetilde{\mathcal{K}} \models \forall X.\, Q_{[X]} \leftrightarrow \widetilde{Q}_{[X]}$ *(that is, $Q_{[X]}$ is determined),*
2. *$Q_{[X]}$ is safe-range (that is, $Q_{[X]}$ is domain independent),*
3. *$\mathcal{K}$ is safe-range (that is, $\mathcal{K}$ is domain independent),*

then there exists an exact reformulation $\widehat{Q}_{[X]}$ of $Q_{[X]}$ as a safe-range query in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ over $\mathbb{P}_D$ under $\mathcal{K}$, and $\widehat{Q}_{[X]}$ can be obtained constructively.

We conclude by mentioning the two decidable description logics $\mathcal{ALCHOI}$ and $\mathcal{ALCHOQ}$, which have a well defined and intuitive syntactic fragment (based on a

notion of "safe-range" similar to the one in first-order logic) characterising exactly their domain independent fragment, and which have finitely controllable determinacy. So, these two logic are excellent candidates to play the role of ontology languages on top of databases represented as DBoxes.

Theorem 5 (Description logics version of Codd's Theorem) *The domain independent fragments of $\mathcal{ALCHOI}$ and $\mathcal{ALCHOQ}$ are equally expressive to the safe-range fragments of respectively $\mathcal{ALCHOI}$ and $\mathcal{ALCHOQ}$ and they have finitely controllable determinacy.*

Acknowledgements This research has been carried out within the Euregio IPN12 KAOS, funded by the "European Region Tyrol-South Tyrol-Trentino" (EGTC) under the first call for basic research projects, and by unibz. It has also been supported by the unibz CRC projects KENDO and OnProm. We wish to thank Volha Kerhet and Nhung Ngo for their crucial contributions to the results presented in this chapter.

References

1. S. Abiteboul, R. Hull, V. Vianu, *Foundations of Databases* (Addison Wesley Publ Co, Reading, 1995)
2. A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009)
3. M. Benedikt, B. ten Cate, E. Tsamoura, Generating low-cost plans from proofs. Proc. PODS **2014**, 200–211 (2014)
4. M. Benedikt, J. Leblay, B. ten Cate, E. Tsamoura, *Generating Plans from Proofs: The Interpolation-based Approach to Query Reformulation*. Synthesis Lectures on Data Management (Morgan & Claypool Publishers, 2016)
5. E. Beth, On Padoa's method in the theory of definition. Indag. Math. **15**, 330–339 (1953)
6. D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: answering SPARQL queries over relational databases. Semantic Web J. **8**(3), 471–487 (2017)
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, Ontologies and databases: the *DL-Lite* approach, in *RW 2009 Tutorial Lectures*, ed. by S. Tessaris, E. Franconi. LNCS, vol. 5689 (Springer, Berlin, 2009)
8. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning **39**(3), 385–429 (2007)
9. B. ten Cate, E. Franconi, İn. Seylan, Beth definability in expressive description logics, in *Proceedings of IJCAI 2011* (2011), pp. 1099–1106
10. B. ten Cate, E. Franconi, I. Seylan, Beth definability in expressive description logics. J. Artif. Intell. Res. **48**, 347–414 (2013)
11. R.A. Di Paola, The recursive unsolvability of the decision problem for the class of definite formulas. J. ACM **16**(2), 324–327 (1969)
12. T. Eiter, M. Ortiz, M. Simkus, T.K. Tran, G. Xiao, Query rewriting for Horn-SHIQ plus rules, in *Proceedings of AAAI 2012* (AAAI Press, 2012), pp. 726–733
13. O. Etzioni, K. Golden, D. Weld, Sound and efficient closed-world reasoning for planning. Artif. Intell. **89**, 113–148 (1996)
14. O. Etzioni, K. Golden, D.S. Weld, Sound and efficient closed-world reasoning for planning. Artif. Intell. **89**, 113–148 (1997)

15. W. Fan, F. Geerts, L. Zheng, View determinacy for preserving selected information in data transformations. Inf. Syst. **37**, 1–12 (2012)
16. I. Feinerer, E. Franconi, P. Guagliardo, Lossless selection views under conditional domain constraints. IEEE Trans. Knowl. Data Eng. **27**(2), 504–517 (2015)
17. E. Franconi, Y.A. Ibanez-Garcia, İ. Seylan, Query answering with DBoxes is hard. ENTCS **278**, 71–84 (2011)
18. E. Franconi, V. Kerhet, N. Ngo, Exact query reformulation with first-order ontologies and databases. Proc. JELIA **2012**, 202–214 (2012)
19. E. Franconi, V. Kerhet, N. Ngo, Exact query reformulation over databases with first-order and description logics ontologies. J. Artif. Intell. Res. **48**, 885–922 (2013)
20. G. Gottlob, S. Kikot, R. Kontchakov, V.V. Podolskii, T. Schwentick, M. Zakharyaschev, The price of query rewriting in ontology-based data access. Artif. Intell. **213**, 42–59 (2014)
21. A.Y. Halevy, Answering queries using views: a survey. VLDB J. **10**, 270–294 (2001)
22. R. Kontchakov, C. Lutz, D. Toman, F. Wolter, M. Zakharyaschev, The combined approach to query answering in DL-Lite. Proc. KR **2010**, 247–257 (2010)
23. A. Krisnadhi, C. Lutz, Data complexity in the $\mathcal{EL}$ family of description logics. Proc. LPAR **2007**, 333–347 (2007)
24. M. Krötzsch, S. Rudolph, P. Hitzler, Complexity boundaries for horn description logics. Proc. AAAI **2007**, 452–457 (2007)
25. M. Marx, Queries determined by views: pack your views. Proc. PODS **2007**, 23–30 (2007)
26. A. Nash, L. Segoufin, V. Vianu, Views and queries: determinacy and rewriting. ACM Trans. Database Syst. **35**, 21:1–21:41 (2010)
27. N. Ngo, E. Franconi, Unique solutions in data exchange under STS mappings, in *Proceedings of AMW 2016* (2016)
28. N. Ngo, M. Ortiz, M. Simkus, Closed predicates in description logics: results on combined complexity. Proc. KR **2016**, 237–246 (2016)
29. H. Pérez-Urbina, B. Motik, I. Horrocks, Tractable query answering and rewriting under description logic constraints. J. Appl. Logic **8**(2), 186–209 (2010)
30. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking data to ontologies. J. Data Semant. X, 133–173 (2008)
31. I. Pratt-Hartmann, Complexity of the two-variable fragment with counting quantifiers. J. Logic Lang. Inf. **14**(3), 369–395 (2005)
32. M. Rodriguez-Muro, D. Calvanese, High performance query answering over DL-Lite ontologies. Proc. KR **2012**, 308–318 (2012)
33. R. Rosati, On conjunctive query answering in $\mathcal{EL}$, in *Proceedings of DL 2007*. CEUR, vol. 250 (2007), pp. 451–458. www.ceur-ws.org
34. R. Rosati, Prexto: query rewriting under extensional constraints in *DL-Lite*, in *Proceedings of ESWC 2012*. LNCS, vol. 7295 (Springer, Berlin, 2012), pp. 360–374
35. R. Rosati, A. Almatelli, Improving query answering over DL-Lite ontologies. Proc. KR **2010**, 290–300 (2010)
36. İ. Seylan, E. Franconi, J. de Bruijn, Effective query rewriting with ontologies over DBoxes, in *Proceedings of IJCAI 2009* (2009), pp. 923–925
37. Toman, D., Weddell, G.: Fundamentals of Physical Design and Query Compilation. Morgan & Claypool Publishers (2011)
38. M.Y. Vardi, The complexity of relational query languages. Proc. STOC **1982**, 137–146 (1982)