Effective Query Answering with Ontologies and DBoxes

Enrico Franconi(✉) and Volha Kerhet

KRDB Research Centre for Knowledge and Data,
Free University of Bozen-Bolzano, Bolzano, Italy
{franconi,kerhet}@inf.unibz.it
http://krdb.eu

Abstract. The goal of this chapter is to survey the formalisation of a precise and uniform integration between first-order ontologies, first-order queries, and classical relational databases (DBoxes) We include here non-standard variants of first-order logic, such as the one with active domain semantics and standard name assumption, used typically in database theory. We present a general framework for the rewriting of a domain independent first-order query in presence of an arbitrary domain independent first-order logic ontology over a signature extending a database signature with additional predicates. The framework supports deciding the existence of a logically equivalent and – given the ontology – safe-range first-order reformulation (called exact reformulation) of a domain independent first-order query in terms of the database signature, and if such a reformulation exists, it provides an effective approach to construct the reformulation based on interpolation using standard theorem proving techniques (i.e., tableau). Since the reformulation is a safe-range formula, it is effectively executable as an SQL query. We finally present an application of the framework with the very expressive $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ description logics ontologies, by providing effective means to compute safe-range first-order exact reformulations of queries.

1 Introduction

We address the problem of query reformulation with expressive ontologies over databases. An ontology provides a conceptual view of the database and it is composed by 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 using only the basic vocabulary of the relational database directly.

In this chapter we study and develop a query rewriting framework applicable to knowledge representation systems where data is stored in a classical finite relational database, in a way that in the literature has been called the *locally-closed world* assumption [12], *exact views* [13,25,26], or *DBox* [16,31]. A DBox is a set of ground atoms which semantically behaves like a database, i.e., the interpretation of the database predicates in the DBox is exactly equal to the database

C. Lutz et al. (Eds.): Baader Festschrift, LNCS 11560, pp. 301–328, 2019.
https://doi.org/10.1007/978-3-030-22102-7_14

relations in any model. The DBox predicates are *closed*, i.e., their extensions are the same in every interpretation, whereas the other predicates in the ontology are *open*, i.e., their extensions may vary among different interpretations. We do not consider here the *open* interpretation for the database predicates (also called *ABox* or *sound views*). In an ABox the interpretation of database predicates contains the database relations and possibly more data coming from the non-database predicates. This notion is less faithful in the representation of a database semantics since it would allow for spurious interpretations of database predicates with additional unwanted tuples not present in the original database.

In our general framework an ontology is a set of first-order formulas, and queries are (possibly open) first-order formulas. Within this setting, the framework provides precise semantic conditions to decide the existence of a safe-range first-order equivalent reformulation of a query in terms of the database signature. It also provides an constructive approach to build the reformulation with sufficient conditions. We are interested in safe-range reformulations of queries because their range-restricted syntax is needed to reduce the original query answering problem to a relational algebra evaluation (e.g., via SQL) over the original database [1]. Our framework points out several conditions on the ontologies and the queries to guarantee the existence of a safe-range reformulation. We show that these conditions are feasible in practice and we also provide an implementable method to ensure their validation. Standard theorem proving techniques can be used to compute the reformulation.

In order to be complete, our framework is applicable to ontologies and queries expressed in any fragment of first-order logic enjoying finitely controllable determinacy [26], a stronger property than the finite model property of the logic. If the employed logic does not enjoy finitely controllable determinacy our approach would become sound but incomplete, but still effectively implementable using standard theorem proving techniques. We have explored non-trivial applications where the framework is complete; in this chapter, the application with $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ ontologies and concept queries is discussed. We show how (i) to check whether the answers to a given query with an ontology are *solely* determined by the extension of the DBox (database) predicates and, if so, (ii) to find an equivalent rewriting of the query in terms of the DBox predicates to allow the use of standard database technology (SQL) for answering the query. This means we benefit from the low computational complexity in the size of the data for answering queries on relational databases. In addition, it is possible to reuse standard techniques of description logics reasoning to find rewritings, such as in the paper by [31].

The query reformulation problem has received strong interest in classical relational database research as well as modern knowledge representation studies. Differently from the mainstream research on query reformulation [21], which is mostly based on perfect or maximally contained rewritings with sound views under relatively inexpressive constraints (see, e.g., the DL-Lite approach in [2]), we focus here on exact rewritings with exact views, since it characterises precisely the query answering problem with ontologies and databases, in the case

when the exact semantics of the database must be preserved. As an example, consider a ground negative query over a given standard relational database; by adding an ontology on top of it, its answer is not supposed to change—since the query uses only the signature of the database and additional constraints are not supposed to change the meaning of the query—whereas if the database were treated as an ABox (sound views) the answer may change in presence of an ontology. This may be important from the application perspective: a DBox preserves the behaviour of the legacy application queries over a relational database. Moreover, by focussing on exact reformulations of *definable* queries (as opposed to considering the certain answer semantics to arbitrary queries, such as in DL-Lite), we guarantee that answers to queries can be subsequently composed in an arbitrary way: this may be important to legacy database applications. A comprehensive summary comparing the ABox- and DBox-based approaches to data representation in description logics appears in [8].

The approach to query reformulation with first-order theories based on exact rewritings 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 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. In a more practical database settings, 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* [32]. The exact rewriting 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 [14]. 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]. Another application of DBoxes has been in the context of constraints representation in ontologies [29].

The chapter is organised as follows: Sect. 2 provides the necessary formal background and definitions; Sect. 3 introduces the notion of a query determined by a database; Sect. 4 introduces a characterisation of the query reformulation problem; in Sects. 5 and 6 the conditions allowing for an effective reformulation are analysed, and a sound and complete algorithm to compute the reformulation is introduced. Finally, we present the case of $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ ontologies in Sect. 7 and conclude in Sect. 8. This chapter extends the work first presented in [17,18].

2 Preliminaries

Let $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ be a classical function-free first-order language with equality over a signature $\Sigma = (\mathbb{C},\mathbb{P})$, where $\mathbb{C}$ is a set of *constants* and $\mathbb{P}$ is a set of *predicates* with associated arities. The arity of a predicate P we denote by $\text{AR}(P)$. In the rest we will refer to an arbitrary fragment of $\mathcal{FOL}(\mathbb{C},\mathbb{P})$, which will be called $\mathcal{L}$. We denote by $\sigma(\phi)$ the signature of the formula ϕ, that is all the predicates and constants occurring in ϕ. We denote with $\mathbb{P}_{\{\phi_1,\ldots,\phi_n\}}$ the set of all predicates occurring in the formulas $\phi_1,\ldots,\phi_n$, with $\mathbb{C}_{\{\phi_1,\ldots,\phi_n\}}$ the set of all constants occurring in the formulas $\phi_1,\ldots,\phi_n$; for the sake of brevity, instead of $\mathbb{P}_{\{\phi\}}$ (resp. $\mathbb{C}_{\{\phi\}}$) we write $\mathbb{P}_\phi$ (resp. $\mathbb{C}_\phi$). We denote with $\sigma(\phi_1,\ldots,\phi_n)$ the signature of the formulas $\phi_1,\ldots,\phi_n$, namely the union of $\mathbb{P}_{\{\phi_1,\ldots,\phi_n\}}$ and $\mathbb{C}_{\{\phi_1,\ldots,\phi_n\}}$. We denote the set of all variables appearing in ϕ as $\text{VAR}(\phi)$, and the set of the free variables appearing in ϕ as $\text{FREE}(\phi)$; we may use for ϕ the notation $\phi(\bar{x})$, where $\bar{x} = \text{FREE}(\phi)$ is the (possibly empty) set of free variables of the formula. The notation $\phi(\bar{x},\bar{y})$ means $\text{FREE}(\phi) = \bar{x} \cup \bar{y}$. A formula in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ is in *prenex normal form*, if it is written as a string of quantifiers followed by a quantifier-free part. Every formula is equivalent to a formula in prenex normal form and can be converted into it in polynomial time [23].

Let $\mathbb{X}$ be a countable set of variables we use. We define a *substitution* Θ to be a total function $\mathbb{X} \mapsto \mathbb{S}$ assigning an element of the set $\mathbb{S}$ to each variable in $\mathbb{X}$. We can see substitution as a countable set of assignments of elements from $\mathbb{S}$ to elements from $\mathbb{X}$. That is, if $\mathbb{X} = \{x_1, x_2, \ldots\}$, then $\Theta := \{x_1 \to s_1, x_2 \to s_2, \ldots\}$, where $s_1, s_2, \ldots$ are elements from $\mathbb{S}$ assigned to corresponding variables from $\mathbb{X}$ by Θ.

As usual, an *interpretation* $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ includes a non-empty set – the domain $\Delta^\mathcal{I}$ – and an interpretation function $\cdot^\mathcal{I}$ defined over constants and predicates of the signature. We say that interpretations $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ and $\mathcal{J} = \langle \Delta^\mathcal{J}, \cdot^\mathcal{J} \rangle$ are *equal*, written $\mathcal{I} = \mathcal{J}$, if $\Delta^\mathcal{I} = \Delta^\mathcal{J}$ and $\cdot^\mathcal{I} = \cdot^\mathcal{J}$. We use standard definitions of validity, satisfiability and entailment of a formula. An *extension* of $\phi(\bar{x})$ in interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, denoted $(\phi(\bar{x}))^\mathcal{I}$, is the set of substitutions from the variable symbols to elements of $\Delta^\mathcal{I}$ which satisfy ϕ in $\mathcal{I}$. That is,

$$(\phi(\bar{x}))^\mathcal{I} = \{\Theta : \mathbb{X} \mapsto \Delta^\mathcal{I} \mid \mathcal{I}, \Theta \models \phi(\bar{x})\}.$$

If ϕ is closed, then the extension depends on whether ϕ holds in $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$ or not. Thus, for a closed formula ϕ, $(\phi)^\mathcal{I} = \{\Theta \mid \Theta : \mathbb{X} \mapsto \Delta^\mathcal{I}\}$ – the set of all possible substitutions assigning elements from the domain $\Delta^\mathcal{I}$ to variables $\mathbb{X}$ – if $\mathcal{I} \models \phi$, and $(\phi)^\mathcal{I} = \emptyset$, if $\mathcal{I} \not\models \phi$.

Given an interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, \cdot^\mathcal{I} \rangle$, we denote by $\mathcal{I}|_\mathbb{S}$ the interpretation restricted to a smaller signature $\mathbb{S} \subseteq \mathbb{P} \cup \mathbb{C}$, i.e., the interpretation with the same domain $\Delta^\mathcal{I}$ and the same interpretation function $\cdot^\mathcal{I}$ defined only for the constants and predicates from the set $\mathbb{S}$. The *semantic active domain of the signature* $\sigma' \subseteq \mathbb{P} \cup \mathbb{C}$ *in an interpretation* $\mathcal{I}$, denoted $adom(\sigma', \mathcal{I})$, is the set of all elements of the domain $\Delta^\mathcal{I}$ occurring in interpretations of predicates and

constants from σ' in $\mathcal{I}$:

$$adom(\sigma', \mathcal{I}) := \bigcup_{P \in \sigma'} \bigcup_{(a_1,\ldots,a_n) \in P^{\mathcal{I}}} \{a_1, \ldots, a_n\} \cup \bigcup_{c \in \sigma'} \{c^{\mathcal{I}}\}.$$

If $\sigma' = \sigma(\phi)$, where ϕ is a formula, we call $adom(\sigma(\phi), \mathcal{I})$ a *semantic active domain of the formula ϕ in an interpretation $\mathcal{I}$*.

Let $X \subseteq \mathbb{X}$ be a set of variables and $\mathbb{S}$ a set. Let us consider the restriction of a substitution to a set of variables from $\mathbb{X}$. That is, we consider a function $\Theta|_X$ assigning an element in $\mathbb{S}$ to each variable in X. We abuse the notation and call such restriction simply *substitution*. Thus, hereafter substitution is a function from a set of variables $X \subseteq \mathbb{X}$ to a set S: $\Theta : X \mapsto S$, including the empty substitution ϵ when $X = \emptyset$. *Domain* and *image* (*range*) of a substitution Θ are written as $dom(\Theta)$ and $rng(\Theta)$ respectively.

Given a subset of the set of constants $\mathbb{C}' \subseteq \mathbb{C}$, we write that a formula $\phi(\bar{x})$ is true in an interpretation $\mathcal{I}$ with its free variables substituted according to a substitution $\Theta : \bar{x} \mapsto \mathbb{C}'$ as $\mathcal{I} \models \phi(\bar{x}/\Theta)$. Given an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and a subset of its domain $\Delta \subseteq \Delta^{\mathcal{I}}$, we write that a formula $\phi(\bar{x})$ is true in $\mathcal{I}$ with its free variables interpreted according to a substitution $\Theta : \bar{x} \mapsto \Delta$ as $\mathcal{I}, \Theta \models \phi(\bar{x})$.

A (possibly empty) finite set $\mathcal{KB}$ of closed formulas will be called an *ontology*. As usual, an interpretation in which a closed formula is true is called a *model* for the formula; the set of all models of a formula ϕ (respectively $\mathcal{KB}$) is denoted by $M(\phi)$ (respectively $M(\mathcal{KB})$).

2.1 DBoxes

A *DBox* $\mathcal{DB}$ is a *finite* set of ground atoms of the form $P(c_1, \ldots, c_n)$, where $P \in \mathbb{P}$, n-ary predicate, and $c_i \in \mathbb{C}$ $(1 \leq i \leq n)$. DBox can be seen as a variant of database representation. The set of all predicates appearing in a DBox $\mathcal{DB}$ is denoted by $\mathbb{P}_{\mathcal{DB}}$, and the set of all constants appearing in $\mathcal{DB}$ is called the *active domain of* $\mathcal{DB}$, and is denoted by $\mathbb{C}_{\mathcal{DB}}$.

An interpretation $\mathcal{I}$ *embeds a DBox* $\mathcal{DB}$, if $a^{\mathcal{I}} = a$ for every DBox constant $a \in \mathbb{C}_{\mathcal{DB}}$ (the *standard name assumption (SNA)*, customary in databases, see [1]) and that, for any $o_1, \ldots, o_n \in \Delta^{\mathcal{I}}$, $(o_1, \ldots, o_n) \in P^{\mathcal{I}}$ if and only if $P(o_1, \ldots, o_n) \in \mathcal{DB}$. We denote the set of all interpretations embedding a DBox $\mathcal{DB}$ as $E(\mathcal{DB})$. A DBox $\mathcal{DB}$ is *legal for an ontology* $\mathcal{KB}$ if there exists a model of $\mathcal{KB}$ embedding $\mathcal{DB}$.

In other words, in every interpretation embedding $\mathcal{DB}$ the interpretation of any DBox predicate is always the same and it is given exactly by its content in the DBox; this is, in general, not the case for the interpretation of the non-DBox predicates. We say that all the DBox predicates are *closed*, while all the other predicates are *open* and may be interpreted differently in different interpretations. We do not consider here the *open world* assumption (the *ABox*) for embedding a DBox in an interpretation. In an open world, an interpretation $\mathcal{I}$ soundly embeds a DBox if it holds that $(c_1, \ldots, c_n) \in P^{\mathcal{I}}$ if (but *not* only if) $P(c_1, \ldots, c_n) \in \mathcal{DB}$.

In order to allow for an arbitrary DBox to be embedded, we generalise the *standard name assumption* to all the constants in $\mathbb{C}$; this implies that the domain of any interpretation necessarily includes the set of all the constants $\mathbb{C}$, which we assume to be finite. The finiteness of $\mathbb{C}$ corresponds to the finite ability of a database system to represent distinct constant symbols; $\mathbb{C}$ is meant to be unknown in advance, since different database systems may have different limits. We will see that the framework introduced here will not depend on the choice of $\mathbb{C}$.

If $\sigma' \subseteq \mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}$, then for any interpretations $\mathcal{I}$ and $\mathcal{J}$ embedding $\mathcal{DB}$ we have: $adom(\sigma', \mathcal{I}) = adom(\sigma', \mathcal{J})$; so, for such a case we introduce the notation $adom(\sigma', \mathcal{DB}) := adom(\sigma', \mathcal{I})$, where $\mathcal{I}$ is any interpretation embedding the DBox $\mathcal{DB}$, and call it a *semantic active domain of the signature σ' in a DBox $\mathcal{DB}$*. Intuitively, $adom(\sigma', \mathcal{DB})$ includes the constants from σ' and from $\mathcal{DB}$ appearing in the relations corresponding to the predicates from σ'. If $\sigma' = \sigma(\phi)$, where ϕ is a formula expressed in terms of only DBox predicates from $\mathbb{P}_{\mathcal{DB}}$ (and possibly some constants), we call $adom(\sigma(\phi), \mathcal{DB})$ a *semantic active domain of the formula ϕ in a DBox $\mathcal{DB}$*.

2.2 Queries

A *query* is a (possibly closed) formula. Given a query $\mathcal{Q}(\bar{x})$. We define the *certain answer to $\mathcal{Q}(\bar{x})$ over a DBox $\mathcal{DB}$ and under an ontology $\mathcal{KB}$* as follows:

Definition 1 (Certain answer). *The (certain) answer to a query $\mathcal{Q}(\bar{x})$ over a DBox $\mathcal{DB}$ under an ontology $\mathcal{KB}$ is the set of substitutions with constants:*

$$\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq \mathbb{C},\ \forall \mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\}.$$

Query answering is defined as an *entailment* problem, and as such it is going to have the same (high) complexity as entailment.

Note that if a query $\mathcal{Q}$ is closed (i.e., a *boolean* query), then the certain answer is $\{\epsilon\}$ if $\mathcal{Q}$ is true in all the models of the ontology embedding the DBox, and $\emptyset$ otherwise. In the following, we assume that the closed formula $\mathcal{Q}(\bar{x}/\Theta)$ is neither a valid fromula nor an inconsistent formula under the ontology $\mathcal{KB}$ – with Θ a substitution $\Theta : \bar{x} \mapsto \mathbb{C}$ assigning to variables *distinct* constants not appearing in $\mathcal{Q}$, nor in $\mathcal{KB}$, nor in $\mathbb{C}_{\mathcal{DB}}$; this assumption is needed in order to avoid trivial reformulations.

One can see that if an ontology is inconsistent or a DBox is illegal for an ontology, then the certain answer to any query over the DBox under the ontology is a set of all possible substitutions. Also, if an ontology is a tautology, we actually have a simple case of query answering over a database (DBox) without an ontology. Thus, we can discard these cases and assume to have only consistent non-tautological ontologies and legal DBoxes.

We now show that we can weaken the standard name assumption for the constants by just assuming *unique names*, without changing the certain answers. As we said before, an interpretation $\mathcal{I}$ satisfies the standard name assumption

if $c^{\mathcal{I}} = c$ for any $c \in \mathbb{C}$. Alternatively, an interpretation $\mathcal{I}$ satisfies the unique name assumption (UNA) if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for any different $a, b \in \mathbb{C}$. We denote the set of all interpretations satisfying the standard name assumption as $I(SNA)$. We denote the set of all interpretations satisfying the unique name assumption as $I(UNA)$.

The following proposition allows us to freely interchange the standard name and the unique name assumptions with interpretations embedding DBoxes. This is of practical advantage, since we can encode the unique name assumption in classical first-order logic reasoners, and many description logics reasoners do support natively the unique name assumption as an extension to OWL.

Proposition 1 (SNA vs UNA). *For any query $\mathcal{Q}(\bar{x})$, ontology $\mathcal{KB}$ and DBox $\mathcal{DB}$,*

$$\{\Theta \mid dom(\Theta) = \bar{x}, rng(\Theta) \subseteq \mathbb{C}, \forall \mathcal{I} \in I(SNA) \cap M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\} =$$

$$\{\Theta \mid dom(\Theta) = \bar{x}, rng(\Theta) \subseteq \mathbb{C}, \forall \mathcal{I} \in I(UNA) \cap M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\}.$$

Since a query can be an arbitrary first-order formula, its answer may depend on the domain, which we do not know in advance. For example, the query $Q(x) = \neg Student(x)$ over the database (DBox) $\{Student(a), Student(b)\}$, with domain $\{a, b, c\}$ has the answer $\{x \to c\}$, while with domain $\{a, b, c, d\}$ has the answer $\{x \to c, x \to d\}$. Therefore, the notion of *domain independent* queries has been introduced in relational databases. Here we adapt the classical definitions [1,3] to our framework: we need a more general version of domain independence, namely domain independence w.r.t an ontology, i.e., restricted to the models of an ontology.

Definition 2 (Domain independence). *A formula $\mathcal{Q}(\bar{x})$ is domain independent with respect to an ontology $\mathcal{KB}$ if and only if for every two models $\mathcal{I}$ and $\mathcal{J}$ of $\mathcal{KB}$ (i.e., $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$) which have the same interpretations for all the predicates and constants, and for every substitution $\Theta : \bar{x} \mapsto \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$ we have:*

$$rng(\Theta) \subseteq \Delta^{\mathcal{I}} \text{ and } \mathcal{I}, \Theta \models \mathcal{Q}(\bar{x}) \quad \text{iff}$$
$$rng(\Theta) \subseteq \Delta^{\mathcal{J}} \text{ and } \mathcal{J}, \Theta \models \mathcal{Q}(\bar{x}).$$

The above definition reduces to the classical definition of domain independence whenever the ontology is empty. A weaker version of domain independence – which is relevant for open formulas – is the following.

Definition 3 (Ground domain independence). *A formula $\mathcal{Q}(\bar{x})$ is ground domain independent if and only if $\mathcal{Q}(\bar{x}/\Theta)$ is domain independent for every substitution $\Theta : \bar{x} \mapsto \mathbb{C}$.*

For example, the formula $\neg P(x)$ is ground domain independent, but it is not domain independent.

The problem of checking whether a FOL formula is domain independent is undecidable [1]. That is why we consider a well known domain independent

syntactic fragment of FOL introduced by Codd, namely the *safe-range* fragment. We recall the formal definition [1] of a safe-range formula. First, a formula should be transformed to a *safe-range normal form*, denoted by SRNF. A formula ϕ in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ can be transformed to $\text{SRNF}(\phi)$ by the following steps [1]:

- Variable substitution: no distinct pair of quantifiers may employ same variable;
- Remove universal quantifiers;
- Remove implications;
- Push negation;
- Flatten 'and's and 'or's.

Definition 4 (Range restriction of a formula). *Range restriction of a formula ϕ in a safe-range normal form, denoted $rr(\phi)$, is a subset of* $\text{FREE}(\phi)$ *or* $\bot$ *recursively defined as follows:*

- $\phi = R(t_1,\ldots,t_n)$*, where each* t_i *is either a variable or a constant:* $rr(\phi)$ *is a set of variables in* $t_1,\ldots,t_n$*;*
- $\phi = (x = c)$ *or* $\phi = (c = x)$*, where* c *is a constant:* $rr(\phi) = \{x\}$*;*
- $\phi = (x = y)$*:* $rr(\phi) = \emptyset$*;*
- $\phi = \phi_1 \wedge \phi_2$*:* $rr(\phi) = rr(\phi_1) \cup rr(\phi_2)$*;*
- $\phi = \phi_1 \vee \phi_2$*:* $rr(\phi) = rr(\phi_1) \cap rr(\phi_2)$*;*
- $\phi = \phi_1 \wedge (x = y)$*:* $rr(\phi) = rr(\phi_1)$ *if* $\{x,y\} \cap rr(\phi_1) = \emptyset$*;* $rr(\phi) = rr(\phi_1) \cup \{x,y\}$ *otherwise;*
- $\phi = \neg\phi_1$*:* $rr(\phi) = \emptyset \cap rr(\phi_1)$*;*
- $\phi = \exists x \phi_1$*:* $rr(\phi) = rr(\phi_1) \setminus \{x\}$ *if* $x \in rr(\phi_1)$*;* $rr(\phi) = \bot$ *otherwise,*

where $\bot \cup Z = \bot \cap Z = \bot \setminus Z = Z \setminus \bot = \bot$ *for any range restriction of a formula* Z*.*

Definition 5 (Safe-range formula). *A formula ϕ in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ is safe-range if and only if* $rr(\text{SRNF}(\phi)) = \text{FREE}(\phi)$*.*

Definition 6 (Ground safe-range formula). *A formula $\mathcal{Q}(\bar{x})$ is ground safe-range if and only if $\mathcal{Q}(\bar{x}/\Theta)$ is safe-range for every substitution $\Theta : \bar{x} \mapsto \mathbb{C}$.*

It was proved in [1] that a safe-range fragment is *equally expressive* to a domain independent fragment; indeed a well-known Codd's theorem states that any safe-range formula is domain independent, and any domain independent formula can be easily transformed into a logically equivalent safe-range formula.

Intuitively, a formula is safe-range if and only if its variables are bounded by positive predicates or equalities. For example, the formula $\neg A(x) \wedge B(x)$ is safe-range, while queries $\neg A(x)$ and $\forall x.\, A(x)$ are not. An ontology $\mathcal{KB}$ is safe-range (domain independent), if every formula in $\mathcal{KB}$ is safe-range (domain independent). The safe-range fragment of first-order logic with the standard name assumption is equally expressive to the relational algebra, which is the core of SQL [1].

3 Determinacy

The certain answer to a query includes all the substitutions which make the query true in *all* the models of the ontology embedding the DBox: so, if a substitution would make the query true only in some model, then it would be discarded from the certain answer. In other words, it may be the case that the answer to the query is not necessarily the same among all the models of the ontology embedding the DBox. In this case, the query is not fully determined by the given source data; indeed, given the DBox, there is some answer which is possible, but not certain. Due to the indeterminacy of the query with respect to the data, the complexity to compute the certain answer in general increases up to the complexity of entailment in the fragment of first-order logic used to represent the ontology. We focus on the case when a query has the same answer over all the models of the ontology embedding the DBox, 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 the process may decrease (see Sect. 4). The *determinacy* of a query w.r.t. a DBox (source database) [13,25,26] has been called *implicit definability* of a formula (the query) from a set of predicates (the DBox predicates) by Beth [7].

Definition 7 (Finite Determinacy or **Implicit Definability).** *A query* $\mathcal{Q}(\bar{x})$ *is (finitely) determined by (or implicitly definable from) the DBox predicates* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}$ *if and only if for any two models* $\mathcal{I}$ *and* $\mathcal{J}$ *of the ontology* $\mathcal{KB}$ *– both with a* finite *interpretation to the DBox predicates* $\mathbb{P}_{\mathcal{DB}}$ *– whenever* $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}}\cup\mathbb{C}} = \mathcal{J}|_{\mathbb{P}_{\mathcal{DB}}\cup\mathbb{C}}$ *then for every substitution* $\Theta : \bar{x} \mapsto \Delta^{\mathcal{I}}$ *we have:* $\mathcal{I}, \Theta \models \mathcal{Q}(\bar{x})$ *if and only if* $\mathcal{J}, \Theta \models \mathcal{Q}(\bar{x})$.

Intuitively, the answer to an implicitly definable query does not depend on the interpretation of non-DBox predicates. Once the DBox 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 an implicitly definable query depends only on the interpretation of the DBox predicates and constants and on the domain (which are fixed). In practice, by focusing on finite determinacy of queries we guarantee that the user can always interpret the answers 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.

In the following 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*. Sometimes it may be the case that we consider ontology in one fragment ($\mathcal{F}_1$) and query in another one ($\mathcal{F}_2$). Then we say that fragment $\mathcal{F}_1$ has *finitely controllable determinacy of queries from fragment* $\mathcal{F}_2$ if for every query expressed in $\mathcal{F}_2$ and for every ontology expressed in $\mathcal{F}_1$ finite determinacy of the query under the ontology coincides with unrestricted determinacy.

We require that whenever a query is finitely determined then it is also determined in unrestricted models (the reverse is trivially true). Indeed, the results we obtained would fail if finite determinacy and unrestricted determinacy do not coincide: it can be shown [20] that Theorem 3 below fails if we consider only models with a finite interpretation of DBox predicates.

Example 1 (Example from database theory). Let $\mathbb{P} = \{P, R, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P, R\}$,

$$\begin{aligned}\mathcal{KB} = \{&\forall x, y, z.\, R(x, y) \wedge R(x, z) \rightarrow y = z,\\ &\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(z, x),\\ &(\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)) \rightarrow (\forall x.\, A(x) \leftrightarrow P(x))\}.\end{aligned}$$

$\mathcal{KB}$ is domain independent. The formula $\forall x, y.\, R(x, y) \rightarrow \exists z.\, R(y, z)$ is entailed from the first two formulas *only* over finite interpretations of R. The query $\mathcal{Q} = A(x)$ is domain independent and finitely determined by P (it is equivalent to $P(x)$ under the models with a finite interpretation of R), but it is not determined by any DBox predicate under models with an unrestricted interpretation of R. The fragment in which $\mathcal{KB}$ and $\mathcal{Q}$ are expressed does not enjoy finitely controllable determinacy.

The next theorem immediately follows from the example above.

Theorem 1. *Domain independent fragment does not have finitely controllable determinacy.*

Let $\mathcal{Q}$ be any formula in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ and $\widetilde{\mathcal{Q}}$ be the formula obtained from it by uniformly replacing every occurrence of each non-DBox predicate P with a new predicate $\widetilde{P}$. We extend this renaming operator $\widetilde{\cdot}$ to any set of formulas in a natural way. One can check whether a query is implicitly definable by using the following theorem.

Theorem 2 (Testing determinacy, [7]). *A query $\mathcal{Q}(\bar{x})$ is implicitly definable from the DBox predicates $\mathbb{P}_{\mathcal{DB}}$ under the ontology $\mathcal{KB}$ if and only if $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall \bar{x}.\, \mathcal{Q}(\bar{x}) \leftrightarrow \widetilde{\mathcal{Q}}(\bar{x})$.*

This theorem means, that the problem of checking whether a query is implicitly definable reduces to the problem of checking entailment in first-order logic.

The *exact reformulation* of a query [26] (also called *explicit definition* by [7]) is a formula logically equivalent to the query which makes use *only* of DBox predicates and constants.

Definition 8 (Exact reformulation or explicit definability). *A query $\mathcal{Q}(\bar{x})$ is explicitly definable from the DBox predicates $\mathbb{P}_{\mathcal{DB}}$ under the ontology $\mathcal{KB}$ if and only if there is some formula $\widehat{\mathcal{Q}}(\bar{x})$ in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$, such that $\mathcal{KB} \models \forall \bar{x}.\, \mathcal{Q}(\bar{x}) \leftrightarrow \widehat{\mathcal{Q}}(\bar{x})$ and $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$. We call this formula $\widehat{\mathcal{Q}}(\bar{x})$ an exact reformulation of $\mathcal{Q}(\bar{x})$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$.*

Determinacy of a query is completely characterised by the existence of an exact reformulation of the query: it is well known that a first-order query is determined by DBox predicates *if and only if* there exists a first-order exact reformulation.

Theorem 3 (Projective Beth definability, [7]). *A query $\mathcal{Q}(\bar{x})$ is implicitly definable from the DBox predicates $\mathbb{P}_{\mathcal{DB}}$ under an ontology $\mathcal{KB}$, if and only if it is explicitly definable as a formula $\widehat{\mathcal{Q}}(\bar{x})$ in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$.*

3.1 Finite Controllability of Determinacy for GNFO

We consider a *guarded negation first-order logic* (GNFO) [4,5] – a fragment of FOL in which all occurrences of negation are *guarded* by an atomic predicate. Formally it consists of all formulas generated by the following recursive definition:

$$\phi ::= \; R(t_1,\ldots,t_n) \mid t_1 = t_2 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \exists x.\, \phi \mid \alpha \wedge \neg\phi \qquad (1)$$

where each t_i is either a variable or a constant, α in $\alpha \wedge \neg\phi$ is an atomic formula (possibly an equality statement) containing all free variables of ϕ. This fragment is "good" in a sense that it is decidable and has *finite model property*, that we use to prove Theorem 4.

Definition 9 (Answer-guarded formula). *A first-order logic formula is answer-guarded if it has a form*

$$Atom(\bar{x}) \wedge \varphi(\bar{x}),$$

where $\varphi(\bar{x})$ is some first-order logic formula and Atom is a predicate which arity is equal to the number of free variables of the formula.

The following theorem holds.

Theorem 4. *GNFO has finitely controllable determinacy of*

- *answer-guarded GNFO queries;*
- *boolean GNFO queries;*
- *GNFO queries with one free variable.*

This result is interesting as it is and also important for us because we consider GNFO subfragments of DLs for application of our query reformulation framework. Queries in these subfragments are either boolean or with one free variable (concept queries) (Sect. 7).

4 Exact Safe-Range Query Reformulation

In this section we analyse the conditions under which the original query answering problem corresponding to an entailment problem can be reduced systematically to a model checking problem of a safe-range formula over the database (e.g., using a database system with SQL).

Given a DBox signature $\mathbb{P}_{\mathcal{DB}}$, an ontology $\mathcal{KB}$, and a query $\mathcal{Q}(\bar{x})$ expressed in some fragment of $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ and determined by the DBox predicates, our goal is to find a safe-range reformulation $\widehat{\mathcal{Q}}(\bar{x})$ of $\mathcal{Q}(\bar{x})$ in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$, that when evaluated as a relational algebra expression over a legal DBox, gives the same answer as the certain answer to $\mathcal{Q}(\bar{x})$ over the DBox under $\mathcal{KB}$. This can be reformulated as the following problem:

Problem 1 (Exact safe-range query reformulation). Find an exact reformulation $\widehat{\mathcal{Q}}(\bar{x})$ of $\mathcal{Q}(\bar{x})$ under $\mathcal{KB}$ as a safe-range query in $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ over $\mathbb{P}_{\mathcal{DB}}$.

Since an exact reformulation is equivalent under the ontology to the original query, the certain answer to the original query and to the reformulated query are identical. More precisely, the following proposition holds.

Proposition 2. *Given a DBox $\mathcal{DB}$, let $\mathcal{Q}(\bar{x})$ be implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ and let $\widehat{\mathcal{Q}}(\bar{x})$ be an exact reformulation of $\mathcal{Q}(\bar{x})$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$, then:*

$$\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq \mathbb{C},\ \forall \mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\} =$$
$$\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq \mathbb{C},\ \forall \mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \widehat{\mathcal{Q}}(\bar{x}/\Theta)\}.$$

From the above equation it is clear that in order to answer to an exactly reformulated query, one may still need to consider all the models of the ontology embedding the DBox, i.e., we still have an entailment problem to solve. The following theorem states the condition to reduce the original query answering problem – based on entailment – to the problem of checking the validity of the exact reformulation over a *single* model: the condition is that the reformulation should be domain independent.

Theorem 5 (Adequacy of exact safe-range query reformulation). *Let $\mathcal{DB}$ be a DBox which is legal for $\mathcal{KB}$, and let $\mathcal{Q}(\bar{x})$ be a query. If $\widehat{\mathcal{Q}}(\bar{x})$ is an exact domain independent (or safe-range) reformulation of $\mathcal{Q}(\bar{x})$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$, then:*

$$\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq \mathbb{C},\ \forall \mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\} =$$
$$\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq adom(\sigma(\widehat{\mathcal{Q}}), \mathcal{DB}), \forall \mathcal{I} = \langle \mathbb{C}, \cdot^{\mathcal{I}} \rangle \in E(\mathcal{DB}) :$$
$$\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}} \models \widehat{\mathcal{Q}}(\bar{x}/\Theta)\}.$$

Since, given a DBox $\mathcal{DB}$, for all interpretations $\mathcal{I} = \langle \mathbb{C}, \cdot^{\mathcal{I}} \rangle$ embedding $\mathcal{DB}$ there is only one interpretation $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}}$ with the signature restricted to the DBox

predicates, this theorem reduces the entailment problem to a model checking problem.

A safe-range reformulation is *necessary* to transform a first-order query to a relational algebra query which can then be evaluated by using SQL techniques. The theorem above shows in addition that being safe-range is also a *sufficient* property for an exact reformulation to be correctly evaluated as an SQL query.

Let us now see an example in which we cannot reduce the problem of answering an exact reformulation to model checking over a DBox, if the exact reformulation is not safe-range.

Example 2. Let $\mathbb{P} = \{P, A\}$, $\mathbb{P}_{\mathcal{DB}} = \{P\}$, $\mathbb{C} = \{a\}$, $\mathcal{DB} = \{P(a,a)\}$, $\mathcal{KB} = \{\forall y.\, P(a,y) \vee A(y)\}$, $\mathcal{Q}(\bar{x}) = \widehat{\mathcal{Q}}(\bar{x}) = \forall y.\, P(x,y)$ (i.e., $\bar{x} = \{x\}$).

- $\mathbb{C}$ includes the active domain $\mathbb{C}_{\mathcal{DB}}$ (it is actually equal).
- $\mathcal{DB}$ is legal for $\mathcal{KB}$ because there is $\mathcal{I} = \langle \{a\}, \cdot^{\mathcal{I}} \rangle$ such that $P^{\mathcal{I}} = \{(a,a)\}$, $A^{\mathcal{I}} = \emptyset$ and obviously, $\mathcal{I} \in M(\mathcal{KB})$.
- $\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq \mathbb{C},\ \forall \mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB}) : \mathcal{I} \models \mathcal{Q}(\bar{x}/\Theta)\} = \emptyset$ because one can take $\mathcal{I} = \langle \{a,b\}, \cdot^{\mathcal{I}} \rangle$ such that $P^{\mathcal{I}} = \{(a,a)\}$, $A^{\mathcal{I}} = \{b\}$; then $\mathcal{I} \in M(\mathcal{KB}) \cap E(\mathcal{DB})$, but for the only possible substitution $\{x \to a\}$ we have: $\mathcal{I} \not\models \forall y\, P(a,y)$.
- However, $\{\Theta \mid dom(\Theta) = \bar{x},\ rng(\Theta) \subseteq adom(\sigma(\widehat{\mathcal{Q}}), \mathcal{DB}),\ \forall \mathcal{I} = \langle \mathbb{C}, \cdot^{\mathcal{I}} \rangle \in E(\mathcal{DB}) : \mathcal{I}|_{\mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}} \models \widehat{\mathcal{Q}}(\bar{x}/\Theta)\} = \{x \to a\}$

As we have seen, answers to a query for which a reformulation exists contain only constants from the active domain of the DBox and the query (Theorem 5); therefore, ground statements in the ontology involving non-DBox predicates and non-active domain constants (for example, as ABox statements) will not play any role in the final evaluation of the reformulated query over the DBox.

5 Conditions for an Exact Safe-Range Reformulation

We have just seen the importance of getting an exact safe-range query reformulation. In this section we are going to study the conditions under which an exact safe-range query reformulation exists.

First of all, we will focus on the semantic notion of safe-range, namely domain independence. While implicit definability is – as we already know – a sufficient condition for the existence of an exact reformulation, it does not guarantee alone the existence of a domain independent reformulation.

Example 3. Let $\mathbb{P} = \{A, B\}$, $\mathbb{P}_{\mathcal{DB}} = \{B\}$, $\mathcal{KB} = \{\forall x.B(x) \leftrightarrow A(x)\}$, $\mathcal{Q}(x) = \neg A(x)$. Then $\mathcal{Q}(x)$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, and every exact reformulation of $\mathcal{Q}(x)$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ is logically equivalent to $\neg A(x)$ and not domain independent.

By looking at the example, it seems that the reason for the non domain independent reformulation lies in the fact that the ontology, which is domain independent, cannot guarantee existence of an exact domain independent reformulation of the non domain independent query. However, let us consider the following examples.

Example 4. Let $\mathbb{P}_{\mathcal{DB}} = \{A, C\}$, $\mathcal{KB} = \{\neg A(a), \forall x.\, A(x) \leftrightarrow B(x)\}$ and let a query $\mathcal{Q}(x) = \exists y\, \neg B(y) \wedge C(x)$. It is easy to see that $\mathcal{KB}$ is domain independent and $\mathcal{Q}(x)$ is not. $\mathcal{Q}(x)$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, and $\widehat{\mathcal{Q}}(x) = \neg A(a) \wedge C(x)$ is an exact domain independent reformulation of $\mathcal{Q}(x)$.

Example 5. Let $\mathbb{P}_{\mathcal{DB}} = \{B\}$, $\mathcal{KB} = \{\neg A(a), \forall x.\, A(x) \leftrightarrow B(x)\}$ and let a query $\mathcal{Q} = \neg A(a)$. $\mathcal{KB}$ and $\mathcal{Q}$ are domain independent. $\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$, and $\widehat{\mathcal{Q}} = \exists y\, \neg B(y)$ is an exact reformulation of $\mathcal{Q}$, which is not domain independent.

That is, the following proposition holds.

Proposition 3. *Domain independence of an ontology and an original query does not guarantee domain independence of an exact reformulation of the query under the ontology over any set of DBox predicates.*

It is obvious that in spite of the fact that the query $\mathcal{Q}(x)$ form the Example 4 is not domain independent, it is domain independent with respect to the ontology $\mathcal{KB}$. In other words, in this case the ontology guarantees the existence of an exact domain independent reformulation. With queries that are domain independent with respect to an ontology, the following theorem holds, giving the *semantic* requirements for the existence of an exact domain independent reformulation.

Theorem 6 (Semantic characterisation). *Given a set of DBox predicates $\mathbb{P}_{\mathcal{DB}}$, a domain independent ontology $\mathcal{KB}$, and a query $\mathcal{Q}(\bar{x})$. A domain independent exact reformulation $\widehat{\mathcal{Q}}(\bar{x})$ of $\mathcal{Q}(\bar{x})$ over $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ exists if and only if $\mathcal{Q}(\bar{x})$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ and it is domain independent with respect to $\mathcal{KB}$.*

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 $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ where finite and unrestricted determinacy coincide (i.e., a fragment with finitely controllability of determinacy), and gives us a constructive way to compute it, if it exists.

Theorem 7 (Constructive). *Given a DBox $\mathcal{DB}$. If:*

1. *$\mathcal{KB}$ is a safe-range ontology (that is, $\mathcal{KB}$ is domain independent),*
2. *$\mathcal{Q}(\bar{x})$ is a safe-range query (that is, $\mathcal{Q}(\bar{x})$ is domain independent),*
3. *$\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall \bar{x}.\, \mathcal{Q}(\bar{x}) \leftrightarrow \widetilde{\mathcal{Q}}(\bar{x})$ (that is, $\mathcal{Q}(\bar{x})$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$),*

then there exists an exact reformulation $\widehat{\mathcal{Q}}(\bar{x})$ *of* $\mathcal{Q}(\bar{x})$ *as a safe-range query in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ *over* $\mathbb{P}_{\mathcal{DB}}$ *under* $\mathcal{KB}$ *that can be obtained constructively.*

In order to constructively compute the exact safe-range query reformulation we use the tableau based method to find the Craig's interpolant [15] to compute $\widehat{\mathcal{Q}}(\bar{x})$ from a validity proof of the implication $(\mathcal{KB} \wedge \mathcal{Q}(\bar{x})) \rightarrow (\widetilde{\mathcal{KB}} \rightarrow \widetilde{\mathcal{Q}}(\bar{x}))$. See Sect. 6 for full details.

Let us now consider a fully worked out example, adapted from the paper by [26].

Example 6. Given: $\mathbb{P} = \{R, V_1, V_2, V_3\}$, $\mathbb{P}_{\mathcal{DB}} = \{V_1, V_2, V_3\}$,

$$\begin{aligned}\mathcal{KB} = \{&\forall x, y.\, V_1(x,y) \leftrightarrow \exists z, v.\, R(z,x) \wedge R(z,v) \wedge R(v,y),\\ &\forall x, y.\, V_2(x,y) \leftrightarrow \exists z.\, R(x,z) \wedge R(z,y),\\ &\forall x, y.\, V_3(x,y) \leftrightarrow \exists z, v.\, R(x,z) \wedge R(z,v) \wedge R(v,y)\},\\ \mathcal{Q}(x,y) = \exists z, v, u.\, &R(z,x) \wedge R(z,v) \wedge R(v,u) \wedge R(u,y).\end{aligned}$$

The conditions of the theorem are satisfied: $\mathcal{Q}(x,y)$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$ (since $\sigma(\mathcal{Q}) \subseteq \mathbb{P}_{\mathcal{DB}}$); $\mathcal{Q}(x,y)$ is safe-range; $\mathcal{KB}$ is safe-range. Therefore, with the tableau method one finds the Craig's interpolant to compute $\widehat{\mathcal{Q}}(x,y)$ from a validity proof of the implication $(\mathcal{KB} \wedge \mathcal{Q}(\bar{x})) \rightarrow (\widetilde{\mathcal{KB}} \rightarrow \widetilde{\mathcal{Q}}(\bar{x}))$ and obtain $\widehat{\mathcal{Q}}(x,y) = \exists z.\, V_1(x,z) \wedge \forall v.\, (V_2(v,z) \rightarrow V_3(v,y))$ – an exact ground safe-range reformulation.

Since the answer to $\widehat{\mathcal{Q}}(x,y)$ is in the semantic active domain of the signature $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}} \cup \mathbb{C}_{\widehat{\mathcal{Q}}}$ in the DBox $\mathcal{DB}$ (it follows from Theorem 5), all fee variables in $\widehat{\mathcal{Q}}(x,y)$ can be "guarded" by some DBox predicates or constants. Note that $\mathcal{Q}(x,y) = \exists z, v, u.\, R(z,x) \wedge R(z,v) \wedge R(v,u) \wedge R(u,y) \equiv^{\mathcal{KB}} \exists z, v.\, R(z,x) \wedge R(z,v) \wedge V_2(v,y) \equiv^{\mathcal{KB}} \mathcal{Q}(x,y) \wedge V_2(v,y)$ (where '$\equiv^{\mathcal{KB}}$' means "logically equivalent with respect to $\mathcal{KB}$"). Then $\mathcal{KB} \models \mathcal{Q}(x,y) \leftrightarrow \widehat{\mathcal{Q}}(x,y) \wedge \exists v.\, V_2(v,y)$. Therefore, $\widehat{\mathcal{Q}}(x,y) \wedge \exists v.\, V_2(v,y) = (\exists z.\, V_1(x,z) \wedge \forall v.\, (V_2(v,z) \rightarrow V_3(v,y))) \wedge \exists v.\, V_2(v,y)$ is an exact safe-range reformulation of $\mathcal{Q}(x,y)$ from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$.

6 Constructing the Safe-Range Reformulation

In this section we introduce a method to compute a safe-range reformulation of an implicitly definable query when conditions in Theorem 7 are satisfied. The method is based on the notion of interpolant introduced by [11].

Definition 10 (Interpolant). *The sentence* χ *in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$ *is an interpolant for the sentence* $\phi \rightarrow \psi$ *in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$*, if all predicate and constant symbols of* χ *are in the set of predicate and constant symbols of both* ϕ *and* ψ*, and both* $\phi \rightarrow \chi$ *and* $\chi \rightarrow \psi$ *are valid sentences in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$*.*

Theorem 8 (Craig's interpolation). *If* $\phi \rightarrow \psi$ *is a valid sentence in* $\mathcal{FOL}(\mathbb{C},\mathbb{P})$*, and neither* ϕ *nor* ψ *are valid, then there exists an interpolant.*

Note that the Beth definability (Theorem 3) and Craig's interpolation theorem do not hold for all fragments of $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$: an interpolant may not always be expressed in the fragment itself, but obviously it is in $\mathcal{FOL}(\mathbb{C}, \mathbb{P})$ (because of Theorem 8).

An interpolant is used to find an exact reformulation of a given implicitly definable query as follows.

Theorem 9 (Interpolant as definition). *Let $\mathcal{Q}(\bar{x})$ be a query with $n \geq 0$ free variables implicitly definable from the DBox predicates $\mathbb{P}_{\mathcal{DB}}$ under the ontology $\mathcal{KB}$. Then, the closed formula with $c_1, \ldots, c_n$ distinct constant symbols in $\mathbb{C}$ not appearing in $\mathcal{KB}$ or $\mathcal{Q}(\bar{x})$:*

$$((\bigwedge \mathcal{KB}) \wedge \mathcal{Q}(\bar{x}/c_1, \ldots, c_n)) \rightarrow ((\bigwedge \widetilde{\mathcal{KB}}) \rightarrow \widetilde{\mathcal{Q}}(\bar{x}/c_1, \ldots, c_n)) \tag{2}$$

is valid, and its interpolant $\widehat{\mathcal{Q}}(c_1, \ldots, c_n/\bar{x})$ is an exact reformulation of $\mathcal{Q}(\bar{x})$ under $\mathcal{KB}$ over $\mathbb{P}_{\mathcal{DB}}$.

Therefore, to find an exact reformulation of an implicitly definable query in terms of DBox predicates it is enough to find an interpolant of the implication (2) and then to substitute all the constants $c_1, \ldots, c_n$ back with the free variables $\bar{x}$ of the original query. An interpolant can be constructed from a validity proof of (2) by using automated theorem proving techniques such as tableau or resolution. In order to guarantee the safe-range property of the reformulation, we use a tableau method as in the book by [15].

6.1 Tableau-Based Method to Compute an Interpolant

In this section we recall in our context the tableau based method to compute an interpolant. This method was described and its correctness was proved in [15].

Assume $\phi \rightarrow \psi$ is valid, therefore $\phi \wedge \neg\psi$ is unsatisfiable. Then there is a closed tableau corresponding to $\phi \wedge \neg\psi$. In order to compute an interpolant from this tableau one needs to modify it to a *biased tableau*.

Definition 11 (Biased tableau). *A biased tableau for formulas $\phi \wedge \neg\psi$ is a tree $T = (V, E)$ where:*

- *V is a set of nodes, each node is labelled by a set of biased formulas. A biased formula is an expression in the form of $L(\varphi)$ or $R(\varphi)$ where φ is a formula. For each node n, $S(n)$ denotes the set of biased formulas labelling n.*
- *The root of the tree is labelled by $\{L(\phi), R(\neg\psi)\}$*
- *E is a set of edges. Given 2 nodes n_1 and n_2, $(n_1, n_2) \in E$ iff there is a biased completion rule from n_1 to n_2. We say there is a biased completion rule from n_1 to n_2 if*
  - *$Y(\mu)$ is the result of applying a rule to $X(\varphi)$, where X and Y refer to L or R (for some rules, there are two possibilities of choosing $Y(\mu)$), and*
  - *$S(n_2) = (S(n_1) \setminus \{X(\varphi)\}) \cup \{Y(\mu)\}$.*

Let C be the set of all constants in the input formulas of the tableau. C^{par} extends C with an infinite set of new constants. A constant is new if it does not occur anywhere in the tableau. With these notations, we have the following rules:

– Propositional rules

Negation rules			$\alpha-$rule	$\beta-$rule
$X(\neg\neg\varphi)$	$X(\neg\top)$	$X(\neg\bot)$	$X(\varphi_1 \wedge \varphi_2)$	$X(\neg(\neg\varphi_1 \wedge \neg\varphi_2))$
$X(\varphi)$	$X(\bot)$	$X(\top)$	$X(\varphi_1)$ $X(\varphi_2)$	$X(\varphi_1) \quad \mid \quad X(\varphi_2)$

– First order rules

$\gamma-$rule	$\sigma-$rule
$X(\forall x.\varphi)$	$X(\exists x.\varphi)$
$X(\varphi(t))$ for any $t \in C^{par}$	$X(\varphi(c))$ for a new constant c

– Equality rules

reflexivity rule	replacement rule
	$X(t = u)$ $Y(\varphi(t))$
$X(\varphi)$	$Y(\varphi(u))$
$X(t = t)$ $t \in C^{par}$ occurs in φ	

A node in the tableau is *closed* if it contains $X(\varphi)$ and $Y(\neg\varphi)$. If a node is closed, no rule is applied. In the other words, it becomes a leaf of the tree. A branch is closed if it contains a closed node and a tableau is closed if all of its branches are closed. Obviously, if the standard tableau for first-order logic is closed then so is the biased tableau and vice versa.

Given a closed biased tableau, the interpolant is computed by applying *interpolant rules*. An interpolant rule is written as $S \xrightarrow{int} I$, where I is a formula and $S = \{L(\phi_1), L(\phi_2), ..., L(\phi_n), R(\psi_1), R(\psi_2), ..., R(\psi_m)\}$.

– Rules for closed branches

r1. $S \cup \{L(\varphi), L(\neg\varphi)\} \xrightarrow{int} \bot$ r2. $S \cup \{R(\varphi), R(\neg\varphi)\} \xrightarrow{int} \top$

r3. $S \cup \{L(\bot)\} \xrightarrow{int} \bot$ r4. $S \cup \{R(\bot)\} \xrightarrow{int} \top$

r5. $S \cup \{L(\varphi), R(\neg\varphi)\} \xrightarrow{int} \varphi$ r6. $S \cup \{R(\varphi), L(\neg\varphi)\} \xrightarrow{int} \neg\varphi$

– Rules for propositional cases

$$\text{p1.}\frac{S \cup \{X(\varphi)\} \xrightarrow{int} I}{S \cup \{X(\neg\neg\varphi)\} \xrightarrow{int} I} \qquad \text{p2.}\frac{S \cup \{X(\top)\} \xrightarrow{int} I}{S \cup \{X(\neg\bot)\} \xrightarrow{int} I}$$

$$\text{p3.}\frac{S \cup \{X(\bot)\} \xrightarrow{int} I}{S \cup \{X(\neg\top)\} \xrightarrow{int} I} \qquad \text{p4.}\frac{S \cup \{X(\varphi_1), X(\varphi_2)\} \xrightarrow{int} I}{S \cup \{X(\varphi_1 \wedge \varphi_2)\} \xrightarrow{int} I}$$

$$\text{p5.}\frac{S \cup \{L(\varphi_1)\} \xrightarrow{int} I_1 \; S \cup \{L(\varphi_2)\} \xrightarrow{int} I_2}{S \cup \{L(\neg(\neg\varphi_1 \wedge \neg\varphi_2))\} \xrightarrow{int} I_1 \vee I_2}$$

$$\text{p6.}\frac{S \cup \{R(\varphi_1)\} \xrightarrow{int} I_1 \; S \cup \{R(\varphi_2)\} \xrightarrow{int} I_2}{S \cup \{R(\neg(\neg\varphi_1 \wedge \neg\varphi_2))\} \xrightarrow{int} I_1 \wedge I_2}$$

– Rules for first order cases:

$$\text{f1.}\frac{S \cup \{X(\varphi(p))\} \xrightarrow{int} I}{S \cup \{X(\exists x.\varphi(x))\} \xrightarrow{int} I}$$ where p is a parameter that does not occur in S or φ

$$\text{f2.}\frac{S \cup \{L(\varphi(c))\} \xrightarrow{int} I}{S \cup \{L(\forall x.\varphi(x))\} \xrightarrow{int} I}$$ if c occurs in $\{\phi_1, ..., \phi_n\}$

$$\text{f3.}\frac{S \cup \{R(\varphi(c))\} \xrightarrow{int} I}{S \cup \{R(\forall x.\varphi(x))\} \xrightarrow{int} I}$$ if c occurs in $\{\psi_1, ..., \psi_m\}$

$$\text{f4.}\frac{S \cup \{L(\varphi(c))\} \xrightarrow{int} I}{S \cup \{L(\forall x.\varphi(x))\} \xrightarrow{int} \forall x.I[c/x]}$$ if c does not occur in $\{\phi_1, ..., \phi_n\}$

$$\text{f5.}\frac{S \cup \{R(\varphi(c))\} \xrightarrow{int} I}{S \cup \{R(\forall x.\varphi(x))\} \xrightarrow{int} \exists x.I[c/x]}$$ if c does not occur in $\{\psi_1, ..., \psi_m\}$

– Rules for equality cases

$$\text{e1.}\frac{S \cup \{X(\varphi(p)), X(t = t)\} \xrightarrow{int} I}{S \cup \{X(\varphi(p))\} \xrightarrow{int} I} \qquad \text{e2.}\frac{S \cup \{X(\varphi(u)), X(t = u)\} \xrightarrow{int} I}{S \cup \{X(\varphi(t)), X(t = u)\} \xrightarrow{int} I}$$

$$\text{e3.}\frac{S \cup \{L(\varphi(u)), R(t = u)\} \xrightarrow{int} I}{S \cup \{L(\varphi(t)), R(t = u)\} \xrightarrow{int} t = u \rightarrow I}$$ if u occurs in $\varphi(t), \psi_1, ..., \psi_m$

$$\text{e4.}\frac{S \cup \{R(\varphi(u)), L(t = u)\} \xrightarrow{int} I}{S \cup \{R(\varphi(t)), L(t = u)\} \xrightarrow{int} t = u \wedge I}$$ if u occurs in $\varphi(t), \psi_1, ..., \psi_m$

$$\text{e5.}\frac{S \cup \{L(\varphi(u)), R(t = u)\} \xrightarrow{int} I}{S \cup \{L(\varphi(t)), R(t = u)\} \xrightarrow{int} I[u/t]}$$ if u does not occur in $\varphi(t), \psi_1, ..., \psi_m$

$$\text{e6.}\frac{S \cup \{R(\varphi(u)), L(t = u)\} \xrightarrow{int} I}{S \cup \{R(\varphi(t)), L(t = u)\} \xrightarrow{int} I[u/t]}$$ if u does not occur in $\varphi(t), \psi_1, ..., \psi_m$

In summary, in order to compute an interpolant of ϕ and ψ, one first need to generate a biased tableaux proof of unsatisfiability of $\phi \wedge \neg\psi$ using biased completion rules and then apply interpolant rules from bottom leaves up to the root. Let us consider an example to demonstrate how the method works.

Example 7. Let $\mathbb{P} = \{S, G, U\}$, $\mathbb{P}_{\mathcal{DB}} = \{S, U\}$,

$$\begin{aligned}\mathcal{KB} = \{&\forall x(S(x) \rightarrow (G(x) \vee U(x)))\\ &\forall x(G(x) \rightarrow S(x))\\ &\forall x(U(x) \rightarrow S(x))\\ &\forall x(G(x) \rightarrow \neg U(x))\}\\ \mathcal{Q}(x) = G(x)\end{aligned}$$

Obviously, $\mathcal{Q}$ is implicitly definable from S and U, since the ontology states that G and U partition S. Now we will follow the tableau method to find its exact reformulation. For compactness, we use the notation S^I instead of $S \xrightarrow{int} I$.

$$\begin{aligned}S_0 = \{&L(\forall x(S(x) \rightarrow (G(x) \vee U(x)))),\\ &L(\forall x(G(x) \rightarrow S(x))),\\ &L(\forall x(U(x) \rightarrow S(x))),\\ &L(\forall x(G(x) \rightarrow \neg U(x))),\\ &L(G(c)),\\ &R(\forall x(S(x) \rightarrow (G_1(x) \vee U(x)))),\\ &R(\forall x(G_1(x) \rightarrow S(x))),\\ &R(\forall x(U(x) \rightarrow S(x))),\\ &R(\forall x(G_1(x) \rightarrow \neg U(x))),\\ &R(\neg G_1(c))\}\end{aligned}$$

By applying the rule for $\forall$ and removing the implication, we have:

$$\begin{aligned}S_1 = \{&L(\neg S(c) \vee G(c) \vee U(c)),\\ &L(\neg G(c) \vee S(c))),\\ &L(\neg U(c) \vee S(c)),\\ &L(\neg G(c) \vee \neg U(c)),\\ &L(G(c)),\\ &R(\neg S(c) \vee G_1(c) \vee U(c)),\\ &R(\neg G_1(c) \vee S(c)),\\ &R(\neg U(c) \vee S(c)),\\ &R(\neg G_1(c) \vee \neg U(c)),\\ &R(\neg G_1(c))\};\end{aligned}$$

and the interpolant of S_1 can be computed as follows:

$$\dfrac{\dfrac{\dfrac{S_4 \cup \{R(\neg S(c))\}^{S(c)} \quad S_4 \cup \{R(U(c))\}^{\neg U(c)}}{S_4 = S_3 \cup \{R(\neg S(c) \vee U(c))\}^{(S(c)\wedge\neg U(c))}} \vee \quad S_3 \cup \{R(G_1(c))\}^{\top}}{S_3 = S_2 \cup \{L(\neg U(c))\}^{(S(c)\wedge\neg U(c))}} B.7 \quad S_2 \cup \{L(\neg G(c))\}^{\perp}}{S_2 = S_1 \cup \{L(S(c))\}^{(S(c)\wedge\neg U(c))}} B.5 \quad S_1 \cup \{L(\neg G(c))\}^{\perp}}{S_1^{(S(c)\wedge\neg U(c))}} B.3$$

Therefore, $S(c) \wedge \neg U(c)$ is the interpolant and $\widehat{\mathcal{Q}}(x) = S(x) \wedge \neg U(x)$ is an exact reformulation of $\mathcal{Q}(x)$.

6.2 A Safe-Range Reformulation

Now we want to show that the reformulation computed by the above tableau based method under the condition of Theorem 7 generates a ground safe-range query.

Theorem 10 (Ground safe-range reformulation). *Let $\mathcal{KB}$ be an ontology, and let $\mathcal{Q}(\bar{x})$ be a query which is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$. If $\mathcal{KB}$ and $\mathcal{Q}(\bar{x})$ are safe-range then a reformulation $\widehat{\mathcal{Q}}(\bar{x})$ obtained using the tableau method described in Sect. 6.1 is ground safe-range.*

In other words, the conditions of Theorem 10 guarantee that all quantified variables in the reformulation are range-restricted. We need to consider now the still unsafe free variables. The theorem below will help us deal with non-range-restricted free variables. Let us first define the *active domain predicate of a signature σ'* as the formula:

$$Adom_{\sigma'}(x) := \bigvee_{P \in \mathbb{P} \cap \sigma'} (\exists x_1, \ldots, x_{\mathrm{AR}(P)-1}. P(x, x_1, \ldots, x_{\mathrm{AR}(P)-1}) \vee \ldots \vee \vee P(x_1, \ldots, x_{\mathrm{AR}(P)-1}, x)) \vee \bigvee_{c \in \mathbb{C} \cap \sigma'} (x = c).$$

If $\sigma' = \sigma(\phi)$, where ϕ is a formula, then instead of $Adom_{\sigma(\phi)}$ we simply write $Adom_\phi$ and call it *active domain predicate of the formula ϕ*.

Theorem 11 (Range of the query). *Let $\mathcal{KB}$ be a domain independent ontology, and let $\mathcal{Q}(x_1, \ldots, x_n)$ be a query which is domain independent with respect to $\mathcal{KB}$. Then*

$$\mathcal{KB} \models \forall x_1, \ldots, x_n. \mathcal{Q}(x_1, \ldots, x_n) \rightarrow Adom_{\mathcal{Q}}(x_1) \wedge \ldots \wedge Adom_{\mathcal{Q}}(x_n).$$

Given a safe-range ontology, a safe-range and implicitly definable query is obviously domain independent with respect to the ontology (by definition). By Theorem 10 there exists a ground safe-range exact reformulation obtained using the tableau method. This reformulation is also domain independent with respect to the ontology (by definition). And then Theorem 11 says that the answer to the

reformulation can only include semantic active domain elements of the reformulation. Therefore, the active domain predicate of the reformulation can be used as a "guard" for free variables which are not bounded by any positive predicate. In this way we obtain a new safe-range reformulation from the ground safe-range one.

Based on Theorems 10 and 11, we propose a complete procedure to construct a safe-range reformulation in Algorithm 1.

Algorithm 1. Safe-range reformulation

Input: a safe-range $\mathcal{KB}$, a safe-range and implicitly definable query $\mathcal{Q}(\bar{x})$.
Output: an exact safe-range reformulation $\widehat{\mathcal{Q}}(\bar{x})$.

1: Compute the interpolant $\widehat{\mathcal{Q}}(\bar{x})$ as in Theorem 9
2: For each free variable x which is not bounded by any positive predicate in $\widehat{\mathcal{Q}}(\bar{x})$ do $\widehat{\mathcal{Q}}(\bar{x}) := \widehat{\mathcal{Q}}(\bar{x}) \wedge Adom_{\widehat{\mathcal{Q}}}(x)$
3: Return $\widehat{\mathcal{Q}}(\bar{x})$

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
$\{o\}$	$\{o^I\} \subseteq \Delta^I$
P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
P^-	$\{(y,x) \mid (x,y) \in P^{\mathcal{I}}\}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\exists R$	$\{x \mid \{y \mid (x,y) \in R^{\mathcal{I}}\} \neq \emptyset\}$
$\exists R.C$	$\{x \mid \{y \mid (x,y) \in R^{\mathcal{I}}\} \cap C^{\mathcal{I}} \neq \emptyset\}$
$\forall R.C$	$\{x \mid \text{if } (x,y) \in R^{\mathcal{I}} \text{ then } y \in C^{\mathcal{I}}\}$

Fig. 1. Syntax and semantics of $\mathcal{ALCHOI}$ concepts and roles

7 The Guarded Negation Fragment of $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$

$\mathcal{ALCHOI}$ is an extension of the description logic $\mathcal{ALC}$ with role hierarchies, individuals and inverse roles: it corresponds to the $\mathcal{SHOI}$ description logic without transitive roles. $\mathcal{ALCHOI}$ without inverse roles and with qualified cardinality

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\{o\}$	$\{o\}^I \subseteq \Delta^I$
$\geq nP$	$\{x \mid \#(\{y \mid (x,y) \in P^{\mathcal{I}}\}) \geq n\}$
$\leq nP$	$\{x \mid \#(\{y \mid (x,y) \in P^{\mathcal{I}}\}) \leq n\}$
$\geq nP.C$	$\{x \mid \#(\{y \mid (x,y) \in P^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \geq n\}$
$\leq nP.C$	$\{x \mid \#(\{y \mid (x,y) \in P^{\mathcal{I}}\} \cap C^{\mathcal{I}}) \leq n\}$

Fig. 2. Syntax and semantics of $\mathcal{SHOQ}$ concepts and roles

restrictions and transitive roles forms the description logic $\mathcal{SHOQ}$. For more details see, e.g., [22]. The syntax and semantics of $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ concept expressions and roles is summarised in the Figs. 1 and 2 respectively, where A is an atomic concept, C and D are concepts, o is an individual name, P is an atomic role, and R is either P or P^-. A TBox in $\mathcal{ALCHOI}$ is a set of concept inclusion axioms $C \sqsubseteq D$ and role inclusion axioms $R \sqsubseteq S$ (where C, D are concepts and R, S are roles) with the usual description logics semantics. A TBox in $\mathcal{SHOQ}$ is defined in the same way without a possibility to express inverse roles and with additional possibility to express transitivity axioms $Trans(P)$.

In this section, we present an application of Theorem 7, by introducing the $\mathcal{ALCHOI}_{GN}$ description logic, the *guarded negation* syntactic fragment of $\mathcal{ALCHOI}$ (Fig. 3), and $\mathcal{SHOQ}_{GN^+}$, the *extended guarded negation* syntactic fragment of $\mathcal{SHOQ}$ (Fig. 4). $\mathcal{ALCHOI}_{GN}$ and $\mathcal{SHOQ}_{GN^+}$ restrict $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ respectively by just prescribing that negated concepts should be *guarded* by some generalised atom – an atomic concept, a nominal, an unqualified existential restriction (for $\mathcal{ALCHOI}$) or an unqualified *atleast* number restriction (for $\mathcal{SHOQ}$), i.e., absolute negation is forbidden. $\mathcal{ALCHOI}_{GN}$ is actually at the intersection of the GNFO fragment and $\mathcal{ALCHOI}$ (by definition).

Each of these fragments has the very important property of *coinciding* (being equally expressive to) with the domain independent and safe-range fragments of the corresponding description logic, therefore providing an excellent candidate language for ontologies and queries satisfying the conditions of Theorem 7.

$$
\begin{aligned}
R &::= P \mid P^- \\
B &::= A \mid \{o\} \mid \exists R \\
C &::= B \mid \exists R.C \mid \exists R.\neg C \mid B \sqcap \neg C \mid C \sqcap D \mid C \sqcup D
\end{aligned}
$$

Fig. 3. Syntax of $\mathcal{ALCHOI}_{GN}$ concepts and roles

$$B ::= A \mid \{o\} \mid \geq nP$$
$$C ::= B \mid \geq nP.C \mid \geq nP.\neg C \mid B \sqcap \neg C \mid C \sqcap D \mid C \sqcup D$$

Fig. 4. Syntax of $\mathcal{SHOQ}_{GN^+}$ concepts

Theorem 12 (Expressive power equivalence). *The domain independent (safe-range) fragment of $\mathcal{ALCHOI}$ and $\mathcal{ALCHOI}_{GN}$ are equally expressive.*

Theorem 13 (Expressive power equivalence). *The domain independent (safe-range) fragment of $\mathcal{SHOQ}$ and $\mathcal{SHOQ}_{GN^+}$ are equally expressive.*

In other words the first theorem says that any domain independent (or safe-range) TBox axiom and any domain independent (or safe-range) concept query in $\mathcal{ALCHOI}$ is logically equivalent, respectively, to a TBox axiom and a concept query in $\mathcal{ALCHOI}_{GN}$, and vice-versa. And the second theorem says that any domain independent (or safe-range) TBox axiom and any domain independent (or safe-range) concept query in $\mathcal{SHOQ}$ is logically equivalent, respectively, to a TBox axiom and a concept query in $\mathcal{SHOQ}_{GN^+}$, and vice-versa.

7.1 Applying the Constructive Theorem

We want to reformulate concept queries over an ontology with a DBox so that the reformulated query can be evaluated as an SQL query over the database represented by the DBox. We consider applications of the Constructive Theorem 7 in the fragments $\mathcal{ALCHOI}_{GN}$ and $\mathcal{SHOQ}_{GN^+}$. In this context, the database is a DBox, the ontology is an $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) TBox, and the query is an $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) *concept query.* A concept query is either an $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) concept expression denoting an open formula with one free variable, or an $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) ABox concept assertion denoting a boolean query.

As expected, a DBox includes ground atomic statements of the form $A(a)$ and $P(a, b)$ (where A is an atomic concept and P is an atomic role). It is easy to prove the following propositions:

Proposition 4. *$\mathcal{ALCHOI}_{GN}$ TBoxes, concept queries are safe-range (domain independent).*

Proposition 5. *$\mathcal{SHOQ}_{GN^+}$ TBoxes, concept queries are safe-range (domain independent).*

We also proved the following theorems.

Theorem 14. *$\mathcal{ALCHOI}_{GN}$ TBoxes have finitely controllable determinacy of concept queries.*

Theorem 15. *$\mathcal{SHOQ}_{GN^+}$ TBoxes have finitely controllable determinacy of concept queries.*

Therefore, we satisfy the conditions of Theorem 7, with a language which is like the very expressive $\mathcal{ALCHOI}$ description logic, but with *guarded* negation. And we also satisfy the conditions of Theorem 7, with a language which is like the very expressive $\mathcal{SHOQ}$ description logic, but with *extended guarded* negation ("extended" here means that cardinality restrictions and transitivity axioms are allowed in $\mathcal{SHOQ}_{GN^+}$ in spite of the fact that they are not expressible in GNFO).

We argue that non-guarded negation should not appear in a cleanly designed ontology, and, if present, should be fixed. Indeed, the use of *absolute* negative information – such as, e.g., in "a non-`male` is a `female`" ($\neg$ `male` $\sqsubseteq$ `female`) – should be discouraged by a clean design methodology, since the subsumer would include *all sorts* of objects in the universe (but the ones of the subsumee type) without any obvious control. Only *guarded* negative information in the subsumee should be allowed – such as in the axiom "a non-`male person` is a `female`" (`person` $\sqcap \neg$ `male` $\sqsubseteq$ `female`).

This observation suggests a fix for non-guarded negations: for every non-guarded negation users will be asked to replace it by a guarded one, where the guard may be an arbitrary atomic concept, or nominal, or unqualified existential restriction (in the case of $\mathcal{ALCHOI}$) or unqualified *atleast* number restriction (in the case of $\mathcal{SHOQ}$). Therefore, the user is asked to make explicit the *type* of that concept, in a way to make it domain independent (i.e. belonging to $\mathcal{ALCHOI}_{GN}$ or $\mathcal{SHOQ}_{GN^+}$). Note that the type could be also a fresh new atomic concept. We believe that the fix we are proposing for $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ is a reasonable one, and would make all $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ ontologies eligible to be used with our framework.

7.2 A Complete Procedure

$\mathcal{ALCHOI}_{GN}$ and $\mathcal{SHOQ}_{GN^+}$ are decidable logics (as a fragments of $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ respectively) and they are feasible applications of our general framework. Given an $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) ontology $\mathcal{KB}$ and a concept query $\mathcal{Q}$ in $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$), we can apply the procedure below to generate a safe-range reformulation over the DBox concepts and roles (based on the constructive theorem, all the conditions of which are satisfied), if it exists.

Note that the procedure for checking determinacy and computing the reformulation could be run in offline mode at compile time. Indeed, it could be run for each atomic concept in the ontology, and store persistently the outcome for each of them if the reformulation has been successful. This pre-computation may be an expensive operation, since – as we have seen – it is based on entailment, but the complexity involves only the size of the ontology and not of the data.

In order to get an idea about the size of the reformulations of concept queries, for the $\mathcal{ALCFI}$ description logic there is a tableau-based algorithm computing explicit definitions of at most double exponential size [9,10]; this algorithm is optimal because it is also shown that the smallest explicit definition of an implicitly defined concept *may* be double exponentially long in the size of the input TBox.

Input: An $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$) TBox $\mathcal{KB}$, a concept query $\mathcal{Q}$ in $\mathcal{ALCHOI}_{GN}$ ($\mathcal{SHOQ}_{GN^+}$), and a DBox signature (DBox atomic concepts and roles).

Output: A safe-range reformulation $\widehat{\mathcal{Q}}$ expressed over the DBox signature.

1: Check the implicit definability of the query $\mathcal{Q}$ by testing if $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \mathcal{Q} \equiv \widetilde{\mathcal{Q}}$ using a standard OWL2 reasoner ($\mathcal{ALCHOI}_{GN}$ and $\mathcal{SHOQ}_{GN^+}$ are sublanguages of OWL2). Continue if this holds.

2: Compute a safe-range reformulation $\widehat{\mathcal{Q}}$ from the tableau proof generated in step 1 (see Section 6). This can be implemented as a simple extension of a standard description logic reasoner even in the presence of the most important optimisation techniques such as semantic branching, absorption, and backjumping as explained by [31] and [9].

Clearly, similarly to DL-Lite reformulations, more research is needed in order to optimise the reformulation step in order to make it practical. However, note that the framework presented here has a clear advantage from the point of view of *conceptual modelling* since implicit definitions (that is, queries) under general TBoxes can be double exponentially more succinct than acyclic concept definitions (that is, explicit queries over the DBox).

The case of query answering with unrestricted description logics and DBoxes, when the rewriting may not be possible at all, has been studied thoroughly by [24,28] regarding both data and combined complexity.

8 Conclusions and Future Work

We have introduced a framework to compute the exact reformulation of first-order queries to a database (DBox) under constraints. We have found the exact conditions which guarantee that a safe-range reformulation exists, and we show that it can be evaluated as an SQL query over the DBox to give the same answer as the original query under the constraints. Non-trivial case studies have been presented in the field of description logics, with the $\mathcal{ALCHOI}$ and $\mathcal{SHOQ}$ languages.

This framework is useful in data exchange-like scenarios, where the target database (made by determined relations) should be materialised as a proper database, over which arbitrary queries should be performed. This is not achieved in a context with non-exact reformulations preserving the certain answers. In our scenario with description logics ontologies reformulations of concept queries are pre-computed offline once. We have shown that our framework works in theory also in the case of arbitrary safe-range first-order queries.

Next, we would like to study optimisations of reformulations. From the practical perspective, since there might be many rewritten queries from one original query, the problem of selecting an optimised query in terms of query evaluation is very important. In fact, one has to take into account which criteria should be used to optimise, such as: the size of the reformulations, the numbers of used predicates, the priority of predicates, the number of relational operators, and

clever usage of duplicates. In this context one may also want to control the process of formula proving to make it produce an optimal reformulation. For instance, using the tableau method, one may prefer one order of expansion rules application to another and, hence, build another interpolant.

Concurrently, we are exploring the problem of *fixing* real ontologies in order to enforce definability when it is known it should be the case [19]. This happens when it is intuitively obvious that the answer of a query can be found from the available data (that is, the query is definable from the database), but the mediating ontology does not entail the definability. We introduce the novel problem of *definability abduction* and we solve it completely in the data exchange scenario.

There is also another interesting open problem about checking that a given DBox is legal with respect to a given ontology. Remember that a DBox $\mathcal{DB}$ is legal for an ontology $\mathcal{KB}$ if there exists a model of $\mathcal{KB}$ embedding $\mathcal{DB}$. This check involves heavy computations for which an optimised algorithm is still unknown: as a matter of fact, the only known method today is to reduce the problem to a satisfiability problem where the DBox is embedded in a TBox using nominals [16]. More research is needed in order to optimise the reasoning with nominals in this special case.

In the case of description logics, we would like to work on extending the theoretical framework with conjunctive queries: we need finitely controllable determinacy with conjunctive queries, which for some description logics seems to follow from the works by [6,30].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR) **36**, 1–69 (2009)
3. Avron, A.: Constructibility and decidability versus domain independence and absoluteness. Theor. Comput. Sci. **394**, 144–158 (2008). https://doi.org/10.1016/j.tcs.2007.12.008. http://dl.acm.org/citation.cfm?id=1351194.1351447
4. Bárány, V., ten Cate, B., Otto, M.: Queries with guarded negation. PVLDB **5**(11), 1328–1339 (2012)
5. Bárány, V., ten Cate, B., Segoufin, L.: Guarded negation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 356–367. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22012-8_28
6. Bárány, V., Gottlob, G., Otto, M.: Querying the guarded fragment. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010), pp. 1–10 (2010)
7. Beth, E.: On Padoa's method in the theory of definition. Indagationes Math. **15**, 330–339 (1953)
8. Calvanese, D., Franconi, E.: First-order ontology mediated database querying via query reformulation. In: Flesca, S., Greco, S., Masciari, E., Saccà, D. (eds.) A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years. SBD, vol. 31, pp. 169–185. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-61893-7_10

9. ten Cate, B., Franconi, E., Seylan, İ.: Beth definability in expressive description logics. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 1099–1106 (2011)
10. ten Cate, B., Franconi, E., Seylan, I.: Beth definability in expressive description logics. J. Artif. Intell. Res. (JAIR) **48**, 347–414 (2013). https://doi.org/10.1613/jair.4057
11. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. J. Symb. Log. **22**(3), 269–285 (1957)
12. Etzioni, O., Golden, K., Weld, D.S.: Sound and efficient closed-world reasoning for planning. Artif. Intell. **89**, 113–148 (1997). https://doi.org/10.1016/S0004-3702(96)00026-4. http://dl.acm.org/citation.cfm?id=249678.249685
13. Fan, W., Geerts, F., Zheng, L.: View determinacy for preserving selected information in data transformations. Inf. Syst. **37**, 1–12 (2012). https://doi.org/10.1016/j.is.2011.09.001
14. Feinerer, I., Franconi, E., Guagliardo, P.: Lossless selection views under conditional domain constraints. IEEE Trans. Knowl. Data Eng. **27**(2), 504–517 (2015). https://doi.org/10.1109/TKDE.2014.2334327
15. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Springer, New York (1996). https://doi.org/10.1007/978-1-4612-2360-3
16. Franconi, E., Ibanez-Garcia, Y.A., Seylan, İ.: Query answering with DBoxes is hard. Electron. Notes Theor. Comput. Sci. **278**, 71–84 (2011)
17. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation with first-order ontologies and databases. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS (LNAI), vol. 7519, pp. 202–214. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33353-8_16
18. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation over databases with first-order and description logics ontologies. J. Artif. Intell. Res. (JAIR) **48**, 885–922 (2013). https://doi.org/10.1613/jair.4058
19. Franconi, E., Ngo, N., Sherkhonov, E.: The definability abduction problem for data exchange. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 217–220. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33203-6_18
20. Gurevich, Y.: Toward logic tailored for computational complexity. In: Börger, E., Oberschelp, W., Richter, M.M., Schinzel, B., Thomas, W. (eds.) Computation and Proof Theory. LNM, vol. 1104, pp. 175–216. Springer, Heidelberg (1984). https://doi.org/10.1007/BFb0099486
21. Halevy, A.Y.: Answering queries using views: a survey. VLDB J. **10**, 270–294 (2001). https://doi.org/10.1007/s007780100054
22. Horrocks, I., Sattler, U.: Ontology reasoning in the SHOQ(D) description logic. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), pp. 199–204 (2001)
23. Kleene, S.C.: Mathematical Logic. Dover, New York (2002)
24. Lutz, C., Seylan, I., Wolter, F.: Ontology-mediated queries with closed predicates. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015, pp. 3120–3126 (2015). http://ijcai.org/Abstract/15/440
25. Marx, M.: Queries determined by views: pack your views. In: Proceedings of the 26th ACM symposium on Principles of Database Systems, PODS 2007, pp. 23–30 (2007). https://doi.org/10.1145/1265530.1265534
26. Nash, A., Segoufin, L., Vianu, V.: Views and queries: determinacy and rewriting. ACM Trans. Database Syst. **35**, 211–2141 (2010). https://doi.org/10.1145/1806907.1806913

27. Ngo, N., Franconi, E.: Unique solutions in data exchange under STS mappings. In: Proceedings of the 10th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW-2016) (2016). http://ceur-ws.org/Vol-1644/paper5.pdf
28. Ngo, N., Ortiz, M., Simkus, M.: Closed predicates in description logics: results on combined complexity. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, 25–29 April 2016, pp. 237–246 (2016). http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12906
29. Patel-Schneider, P.F., Franconi, E.: Ontology constraints in incomplete and complete data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 444–459. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_28
30. Rosati, R.: On the finite controllability of conjunctive query answering in databases under open-world assumption. J. Comput. Syst. Sci. **77**(3), 572–594 (2011)
31. Seylan, İ., Franconi, E., de Bruijn, J.: Effective query rewriting with ontologies over DBoxes. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 923–925 (2009)
32. Toman, D., Weddell, G.E.: Fundamentals of physical design and query compilation. Synth. Lect. Data Manag. **3**, 1–124 (2011). https://doi.org/10.2200/S00363ED1V01Y201105DTM018