# On Finding Query Rewritings under Expressive Constraints

Alex Borgida[1], Jos de Bruijn[2], Enrico Franconi[2], İnanç Seylan[2], Umberto Straccia[3], David Toman[4], and Grant Weddell[4]

[1] Rutgers University, USA, borgida@cs.rutgers.edu
[2] Free University of Bozen-Bolzano, Italy, *lastname*@inf.unibz.it
[3] ISTI-CNR, Italy, umberto.straccia@isti.cnr.it
[4] University of Waterloo, Canada, {david | gweddell}@cs.uwaterloo.ca

**Abstract** We study a general framework for query rewriting in the presence of general FOL constraints, where standard theorem proving techniques (e.g., tableau or resolution) can be used. The framework supports deciding the existence of an equivalent first-order reformulation of a query in terms of a selected set of database predicates, and if so, it provides an effective approach to constructing such a reformulation based on interpolation. The reformulation is effectively executable as a SQL query, i.e., it is a range-restricted reformulation.

## 1 Introduction

Query reformulation under constraints is central in several areas, including: query optimisation in database systems; establishing connections between conceptual data models and their logical realisations; and data integration.

The paper starts by explicating a framework that supports deciding the existence of an equivalent reformulation of a query in terms of a selected set of predicates—called the *database predicates*, and if so, provides an effective approach to constructing such a reformulation. It is particularly concerned with applying this framework to finding *effectively executable first-order reformulations* a la SQL. The contributions of this paper are as follows:

- We propose a general framework for finding equivalent query rewritings under expressive constraints in the context of combined knowledge and data bases. We then apply this framework to three contexts.
- We provide general properties for the FOL constraints and view definitions that guarantee the resulting rewritten query will be domain independent, and hence can be executed using standard relational engines.
- Concerning conjunctive queries, we partially resolve an open problem posed by Nash *et. al.* [13]: Is it decidable whether there is a rewriting of a conjunctive query over connected conjunctive views?
- We prove in [6] that such rewritings can then be effectively obtained using standard proof techniques for first-order logic which terminate in this case, generalising the procedure proposed in [13].

– We show an application of the framework for constraints expressed in the guarded fragment of first-order logic (which is a generalisation of standard ontology languages), and queries over databases for it.

In addition, we show that this technique generalises to an arbitrary class of first-order constraints, albeit termination is sacrificed, and develop general conditions under which a decision procedure exists (based on the decidability of the underlying constraints).

The rest of the paper is organised as follows: Section 2 provides the necessary background and definitions, Section 3 introduces the framework for query reformulation in the context of both data and knowledge bases, and Sections 4.1 and 4.2 instantiate the general framework to the class of connected conjunctive queries and views, and to standard ontology languages to illustrate the features of the framework. We conclude with future directions of research and open problems. In the Appendix we outline how standard proof techniques can be utilised for generating query reformulations, and we show a fully worked out example.

## 2 Database querying: definitions

Given a signature of (possibly infinitely many) *constants* $\mathbb{C}$ and of *database predicates* $\mathbb{P}_{\mathcal{DB}}$, a *database (instance)* $\mathcal{DB}$ is a set of database predicate-labelled tuples of the form $P : \langle a_1, \ldots, a_n \rangle$, where $P \in \mathbb{P}_{\mathcal{DB}}$ is an $n$-ary database predicate and the $a_i \in \mathbb{C}$ are constants. We denote the set of database constants actually appearing in a database $\mathcal{DB}$ (i.e., the so-called *active domain* of $\mathcal{DB}$) as $\mathbb{C}_{\mathcal{DB}}$.

Let $\mathcal{FOL}$ be a function-free first-order language over a signature extending the above: with the same constants $\mathbb{C}$ and with predicates $\mathbb{P} \supseteq \mathbb{P}_{\mathcal{DB}}$; think of these predicates as possibly helping to define views or constraints. A (possibly empty) finite set $\mathcal{KB}$ of closed formulas in $\mathcal{FOL}$ will be called *constraints* (or a *knowledge base*). An *interpretation* $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ (the *domain*) is a non-empty set, and $\cdot^{\mathcal{I}}$ is a function that maps constants to elements of $\Delta^{\mathcal{I}}$ and $n$-ary predicates ($n \geq 1$) to subsets of $(\Delta^{\mathcal{I}})^n$. We follow the usual inductive definition of $\mathcal{I} \models \varphi$ to denote the model $\mathcal{I}$ of a closed formula $\varphi$, and we denote the set of all models of a formula as $\mathsf{M}(\varphi)$. In case we want to emphasise the domain of an interpretation $\mathcal{I}$ in its symbol, we will use $\mathcal{I}^{(\Delta^{\mathcal{I}})}$. With $\sigma(\varphi_1, \ldots, \varphi_n)$ we denote the set of all predicates occurring in each formula $\varphi_i$, which we call the *signature* of the set of formulas.

We define a substitution $\Theta^{\mathbb{S}}_{\mathbb{X}}$ to be a total function $\mathbb{X} \mapsto \mathbb{S}$ assigning an element of the set $\mathbb{S}$ to each variable symbol in $\mathbb{X}$, including the empty substitution $\epsilon$ when $\mathbb{X} = \emptyset$. As usual, given an interpretation $\mathcal{I}$, a subset of the domain $\Delta \subseteq \Delta^{\mathcal{I}}$, a (possibly closed) formula $\varphi$, the (possibly empty) set $\mathbb{X}$ of all free variables of $\varphi$, a substitution $\Theta^{\Delta}_{\mathbb{X}}$, we say that $\mathcal{I}, \Theta^{\Delta}_{\mathbb{X}} \models \varphi$ iff $\varphi$ is true in $\mathcal{I}$ with its free variables interpreted as assigned by $\Theta^{\Delta}_{\mathbb{X}}$. On the other hand, given a substitution $\Theta^{\mathbb{C}}_{\mathbb{X}}$, we say that $\mathcal{I} \models \varphi_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}$ iff $\varphi$ is true in $\mathcal{I}$ after its free variables have been substituted with constants as specified by $\Theta^{\mathbb{C}}_{\mathbb{X}}$.

We now connect the notion of a database with the notion of an interpretation. An interpretation $\mathcal{I}$ *embeds* a database $\mathcal{DB}$, written $\mathcal{I}_{(\mathcal{DB})}$, iff for every database constant $a_i \in \mathbb{C}_{\mathcal{DB}}$ it holds that $a_i^{\mathcal{I}} = a_i$, and for every $n$-ary database predicate

$P \in \mathbb{P}_{\mathcal{DB}}$ it holds that $\langle a_1, \ldots, a_n \rangle \in P^{\mathcal{I}}$ iff $P : \langle a_1, \ldots, a_n \rangle \in \mathcal{DB}$. In other words, in every interpretation embedding a $\mathcal{DB}$, the interpretation of a database predicate is given exactly by its contents in the database; on the other hand, this is not true for the interpretation of the non-database predicates in $\mathbb{P} \setminus \mathbb{P}_{\mathcal{DB}}$, which may vary across the interpretations embedding the database. Note that the domain $\Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}_{(\mathcal{DB})}$ embedding a database $\mathcal{DB}$ is not fixed, but it necessarily includes the active domain $\mathbb{C}_{\mathcal{DB}}$. We say that a database $\mathcal{DB}$ satisfies the constraints $\mathcal{KB}$ (or a database $\mathcal{DB}$ is consistent with respect to the constraints $\mathcal{KB}$) iff there exists $\mathcal{I}_{(\mathcal{DB})} : \mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB})$; in the following we will consider only consistent databases.

Let a *query* $\mathcal{Q}$ be a (possibly closed) formula in $\mathcal{FOL}$ (possibly with predicate symbols other than ones in $\mathbb{P}_{\mathcal{DB}}$) and $\mathbb{X}$ be the (possibly empty) set of all free variables of $\mathcal{Q}$; we write this as $\mathcal{Q}_{[\mathbb{X}]}$. Intuitively, we can think of a substitution as one possible answer to a query, as specified formally in the following.

**Definition 1 (Query answering).** *The (certain)* answer *of a* query $\mathcal{Q}_{[\mathbb{X}]}$ *to a* database $\mathcal{DB}$ *under* constraints $\mathcal{KB}$ *is a set of substitutions* $\Theta^{\mathbb{C}}_{\mathbb{X}}$ *such that:*

$$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \textit{for every } \mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\}.$$

We focus in this paper on situations where such answers can be computed.
This definition generalises to full FOL constraints a framework which has been called also *locally closed world* in AI (see, e.g., [9]), *exact views* in databases (see, e.g., [1,8,13]), or *DBox* in KR (see [14]).

*Example 1.* Given the unary database predicates `MEETING` and `ACTIVITY`, the constraints $\forall x.\texttt{Project}(x) \rightarrow \texttt{ACTIVITY}(x)$, $\forall x.\texttt{MEETING}(x) \rightarrow \texttt{ACTIVITY}(x)$, $\forall x.\texttt{ACTIVITY}(x) \rightarrow \texttt{Project}(x) \vee \texttt{MEETING}(x)$, and finally $\forall x.\texttt{Project}(x) \rightarrow \neg\texttt{MEETING}(x)$, together with the consistent database $\texttt{ACTIVITY}{:}\langle \texttt{m} \rangle$, $\texttt{ACTIVITY}{:}\langle \texttt{p} \rangle$, and $\texttt{MEETING}{:}\langle \texttt{m} \rangle$, the query $\texttt{Project}(x)$ has the answer $\{x \mapsto \texttt{p}\}$. □

Important special well-behaved queries are the *domain independent* queries; here we adopt a generalisation of the definition in [3].

**Definition 2 (Domain independence).** *A query* $\mathcal{Q}_{[\mathbb{X}]}$ *is* domain independent *iff for every pair of interpretations* $\mathcal{I}$, $\mathcal{J}$ *which agree on the interpretation of the predicates and constants (i.e.,* $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}}$*), and for every substitution* $\Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}}$*:*

$$\textit{act-range}(\Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}}) \subseteq \Delta^{\mathcal{I}} \quad \textit{and} \quad \mathcal{I}, \Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}} \models \mathcal{Q}_{[\mathbb{X}]} \qquad \textit{iff}$$
$$\textit{act-range}(\Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}}) \subseteq \Delta^{\mathcal{J}} \quad \textit{and} \quad \mathcal{J}, \Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}} \models \mathcal{Q}_{[\mathbb{X}]}.$$

**Definition 3 (Ground domain independence).** *A query* $\mathcal{Q}_{[\mathbb{X}]}$ *is* ground domain independent *iff* $\mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}$ *is domain independent for every substitution* $\Theta^{\mathbb{C}}_{\mathbb{X}}$*.*

Ground domain independence is a weaker property than domain independence for a query; we will show that it still makes queries behaving well.

*Example 2.* The query $\texttt{Project}(x)$ is domain independent (and therefore also ground domain independent), the query $\neg\texttt{Project}(x)$ is not domain independent but it is ground domain independent, and the query $\neg\texttt{Project}(x) \wedge \forall y.\texttt{ACTIVITY}(y)$ is neither domain independent nor ground domain independent. □

Let's see how the definitions above work in the special case of classical relational databases, where the signature is restricted to database predicates only ($\mathbb{P} = \mathbb{P}_{\mathcal{DB}}$), the constraints and the query are only over this restricted signature, the database is consistent with respect to the constraints, and the queries are domain independent. We can prove (via the Lemma 2) that when all these conditions are met then the original query answering problem in Definition 1 is reduced to:

$$\{\Theta_{\mathbb{X}}^{\mathbb{C}_{\mathcal{DB}} \cup \mathbb{C}_{\mathcal{Q}}} \mid \mathcal{I}_{(\mathcal{DB})}^{(\mathbb{C}_{\mathcal{DB}} \cup \mathbb{C}_{\mathcal{Q}})} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}_{\mathcal{DB}}}]}\}$$

where $\mathbb{C}_{\mathcal{Q}}$ is the set including the interpretation of each constant $a_i$ appearing in the query $\mathcal{Q}$ but not in $\mathbb{C}_{\mathcal{DB}}$, such that $a_i^{\mathcal{I}} = a_i$. Namely, query answering is reduced to a model checking problem over the unique interpretation $\mathcal{I}_{(\mathcal{DB})}^{(\mathbb{C}_{\mathcal{DB}} \cup \mathbb{C}_{\mathcal{Q}}^{\mathcal{I}})}$, for each $n$-tuple of constants in the active domain together with the constants mentioned in the query, and the constraints do not play any role. Therefore it is enough to evaluate the domain independent query $\mathcal{Q}_{[\mathbb{X}]}$ over the database $\mathcal{DB}$ using standard SQL query evaluation technologies, which is exactly what we expected. That is the reason why its data complexity is as fast as in $\mathsf{AC}^0$ – which is the complexity of model checking first order logic formulas with respect to the size of the interpretation.

*Example 3.* Given the unary database predicates `MEETING` and `ACTIVITY`, the constraint $\forall x.\texttt{MEETING}(x) \rightarrow \texttt{ACTIVITY}(x)$, and the consistent database $\texttt{ACTIVITY}{:}\langle \texttt{m} \rangle$, $\texttt{ACTIVITY}{:}\langle \texttt{p} \rangle$, $\texttt{MEETING}{:}\langle \texttt{m} \rangle$, the domain independent query $\texttt{ACTIVITY}(x) \wedge \neg \texttt{MEETING}(x)$ has the answer $\{x \mapsto \texttt{p}\}$. □

## 3 A general framework for query rewriting

We introduce in this Section implicit and explicit *definability* for queries, and discuss how *explicit* definitions can be used for query rewriting, similar in spirit to the recent work in [12,13].

In the following, given a set of constraints $\mathcal{KB}$, a database $\mathcal{DB}$, and a query $\mathcal{Q}$, for every non-database predicate $P$ in $\sigma(\mathcal{KB}, \mathcal{Q}) \setminus \mathbb{P}_{\mathcal{DB}}$, let $\widetilde{P}$ be a renaming of $P$ – a distinct non-database predicate symbol not appearing anywhere in $\sigma(\mathcal{KB}, \mathcal{Q}) \cup \mathbb{P}_{\mathcal{DB}}$; and let extend the renaming transformation $\widetilde{(\cdot)}$ in the obvious way to formulas and sets of constraints and databases, by renaming the predicates in them. Given an interpretation $\mathcal{I}$, let $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}}}$ be an interpretation with the same domain $\Delta^{\mathcal{I}}$ and with the same interpretation function $\cdot^{\mathcal{I}}$ but defined only for the database predicates $\mathbb{P}_{\mathcal{DB}}$.

**Definition 4 (Implicit definability).** *Let $\mathcal{I}$ and $\mathcal{J}$ be two models of the constraints $\mathcal{KB}$. A query $\mathcal{Q}_{[\mathbb{X}]}$ is* implicitly definable *from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$ iff $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}}} = \mathcal{J}|_{\mathbb{P}_{\mathcal{DB}}}$ and $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$ implies that for every $\Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}}$: $\mathcal{I}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}} \models \mathcal{Q}_{[\mathbb{X}]}$ iff $\mathcal{J}, \Theta_{\mathbb{X}}^{\Delta^{\mathcal{I}}} \models \mathcal{Q}_{[\mathbb{X}]}$.*

$\mathcal{Q}$ is implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ in $\mathcal{KB}$ if any two models of $\mathcal{KB}$ that have the same domain and agree in what they assign to the database predicates in $\mathbb{P}_{\mathcal{DB}}$ also agree in what they assign to $\mathcal{Q}$. In other words, given a set of constraints,

a query $\mathcal{Q}$ is implicitly definable if its extension depends only on the extension of the database predicates; remember that both the constraints and the query may include more predicates than just the database predicates. So, if a query is implicitly definable, then its evaluation depends only on the database predicates, and vice-versa; therefore implicitly definable queries characterise exactly *views*. The class of implicitly definable queries (first introduced by Tarski [15]) is exactly the class of queries for which we will find exact rewritings (i.e., their explicit definitions).

*Example 4 (continues Example 1).* Given the database predicates `MEETING` and `ACTIVITY`, and the constraints introduced in Example 1, the query $\texttt{Project}(x)$ is implicitly definable from the database predicates `MEETING` and `ACTIVITY`. □

An alternative *equivalent* reformulation of the *semantic* definition of implicit definability is the following *syntactic* definition [15].

**Definition 5 (Implicit definability: syntactic).** *A query $\mathcal{Q}_{[\mathbb{X}]}$ is* implicitly definable *from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$ iff $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall \mathbb{X}.\mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}_{[\mathbb{X}]}}$.*

According to this definition, checking whether a query is implicit definable can be reduced to checking a standard entailment in $\mathcal{FOL}$. Note that the above definition holds for unrestricted (i.e., either finite or infinite) models. It is possible that a query is not implicitly definable by considering unrestricted models, while it would be by considering the interpretation of the database predicates to be necessarily finite (i.e., when the database is a *finite* set of tuples). That's why we did not explicitly require the database $\mathcal{DB}$ to be finite. If we require the database $\mathcal{DB}$ to be finite, then the framework proposed in this paper may be incomplete, namely it may miss some rewritings. Note that, for example, the framework would be complete again in the case of the fragment of $\mathcal{FOL}$ considered in Section 4.2, since in the guarded fragment of $\mathcal{FOL}$ unrestricted models entailment and finite models entailment coincide [10]. Moreover, in some cases finite model entailment is undecidable while unrestricted models entailment is decidable (e.g., FD + UID + coverage + disjointness) and then the tradeoff is whether to enforce all the constraints in the family sacrificing finite consequences or whether to restrict the allowed constraints.

If a query is implicitly definable from the database predicates, we could hope to find an equivalent formula using only database predicates, namely its *explicit definition*. This can be seen as the explicit view definition over the database predicates associated to the query. Beth [5] and Craig [7] showed that if a formula $\mathcal{Q}$ is implicitly definable from predicates $\mathbb{P}_{\mathcal{DB}}$ under constraints $\mathcal{KB}$, then there exists its *explicit definition*, i.e., a formula $\widehat{\mathcal{Q}}$ in $\mathcal{FOL}$ with $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$ logically equivalent to $\mathcal{Q}$ given $\mathcal{KB}$.

Note that the notion of explicit definability is based on logical equivalence – that is, over all possible substitutions with elements of the domain – and it is therefore stronger than what is needed to preserve equivalence of query answering as defined in Definition 1 – which is based on substitutions with constants. It would be possible to change the definition of query answering as a

set of substitutions over domain elements as opposed to a set of substitutions over constants. With this change, whenever an equivalent query rewriting exists then an explicit definition would exist. However, such a changed definition of query answering would require all the constants in $\mathcal{FOL}$ to satisfy the *standard name* assumption like active domain elements, i.e., for every constant $a \in \mathbb{C}$ it holds that $a^{\mathcal{I}} = a$. We are again faced with a tradeoff between the expressivity of the constraints (allowing for constants with standard FOL semantics as opposed to constants with standard name assumption) and the completeness of the rewriting process. Of course, if the constraints do not mention constants at all, then the framework would be complete. Note that the framework is complete in the case of the fragment of $\mathcal{FOL}$ considered in Section 4.1, since in this fragment of $\mathcal{FOL}$ constants do not appear in the constraints.

**Definition 6 (Explicit definability).** *A query $\mathcal{Q}_{[\mathbb{X}]}$ is* explicitly definable *from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$ iff there is some formula $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ in $\mathcal{FOL}$ such that $\mathcal{KB} \models \forall \mathbb{X}.\mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widehat{\mathcal{Q}}_{[\mathbb{X}]}$ and $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$.*

**Theorem 1 (Projective Beth definability).** *If a query $\mathcal{Q}$ is implicitly definable from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under constraints $\mathcal{KB}$, then it is explicitly definable as a formula $\widehat{\mathcal{Q}}$ in $\mathcal{FOL}$ with $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$.*

Craig [7] gave to this theorem a constructive proof, based on *interpolation*.

**Lemma 1 (Craig's interpolation).** *If $\varphi \rightarrow \psi$ is a valid formula in $\mathcal{FOL}$ and neither $\varphi$ nor $\psi$ are valid, then there is a formula $\chi$ in $\mathcal{FOL}$, called* interpolant*, whose predicate and constant symbols are among the predicate and constant symbols of both $\varphi$ and $\psi$, and both $\varphi \rightarrow \chi$ and $\chi \rightarrow \psi$ are valid formulas.*

Note that the Beth definability and Craig interpolation theorems do not hold for all fragments of $\mathcal{FOL}$: there may not always be an explicit definition or an interpolant in the fragment itself, but of course there will be one in $\mathcal{FOL}$.

An interpolant is used to find the explicit definition of a given implicitly definable predicate as follows.

**Theorem 2 (Interpolant as definition).** *Let $\mathcal{Q}$ be a query with $n \geq 0$ free variables implicitly defined from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$. Then, the closed formula with $c_1, \ldots, c_n$ new distinct constant symbols in $\mathbb{C}$ not appearing in $\mathcal{KB}$ or $\mathcal{Q}$*

$((\bigwedge \mathcal{KB}) \wedge \mathcal{Q}_{[\mathbb{X}/c_1,\ldots,c_n]}) \rightarrow ((\bigwedge \widetilde{\mathcal{KB}}) \rightarrow \widetilde{\mathcal{Q}}_{[\mathbb{X}/c_1,\ldots,c_n]})$

*is valid, and its interpolant $\widehat{\mathcal{Q}}_{[c_1,\ldots,c_n/\mathbb{X}]}$ defines the query $\mathcal{Q}$ explicitly from the database predicates $\mathbb{P}_{\mathcal{DB}}$ under the constraints $\mathcal{KB}$.*

We combine these results to reduce answering of a definable query to a database under constraints to answering of a rewritten query to the database only.

**Theorem 3 (Query rewriting).** *Let $\mathcal{DB}$ be a database satisfying a set of constraints $\mathcal{KB}$, and $\mathcal{Q}_{[\mathbb{X}]}$ be implicitly definable from $\mathbb{P}_{\mathcal{DB}}$ under $\mathcal{KB}$. Then there is a* rewritten query *$\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ (from Theorem 2) in $\mathcal{FOL}$ with $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$ such that:*

$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid$ *for every* $\mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} =$

$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid$ *for every* $\Delta^{\mathcal{I}} : \mathcal{I}^{(\Delta^{\mathcal{I}})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\}$.

*Proof (Proof (sketch)).* We can replace $\mathcal{Q}$ with $\widehat{\mathcal{Q}}$ since they are logically equivalent in all the models of $\mathcal{KB}$. Since $\widehat{\mathcal{Q}}$ mentions only database predicates, we can restrict the interpretation $\mathcal{I}_{(\mathcal{DB})}$ only to the database predicates, and across these interpretations $\mathcal{I}_{(\mathcal{DB})}$ is always the same but for the domain $\Delta^{\mathcal{I}}$– which necessarily includes the active domain $\mathbb{C}_{\mathcal{DB}}$ – and for the interpretation of constants in $\mathbb{C}$ not in the active domain $\mathbb{C}_{\mathcal{DB}}$.

Even if this theorem shows how to get rid of the constraints $\mathcal{KB}$ by rewriting the original query $\mathcal{Q}$ into a rewritten query $\widehat{\mathcal{Q}}$ over only the database predicates, the original query answering problem in Definition 1 is still not yet reduced to model checking, since we have to consider all possible varying domains extending the active domain. In order to enable the use of standard SQL query evaluation technologies it is necessary to show the *domain independence* of the rewriting $\widehat{\mathcal{Q}}$. Indeed, once we show its domain independence, then we can prove that the original query answering problem in Definition 1 is reducible to the following model checking problem in $\mathcal{FOL}$, for each $n$-tuple of constants in the active domain, and therefore it is enough to evaluate the query $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ over the database $\mathcal{DB}$ using standard SQL query evaluation technologies.

**Lemma 2 (Active domain interpretation).** *Given a database $\mathcal{DB}$ and a domain independent query $\widehat{\mathcal{Q}}$ over $\mathbb{P}_{\mathcal{DB}}$, then:*
$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \textit{for every } \Delta^{\mathcal{I}} : \mathcal{I}^{(\Delta^{\mathcal{I}})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} = \{\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}} \mid \mathcal{I}^{(\mathbb{C}_{\mathcal{DB}})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}}]}\}$,
*where the interpretation of the constants in the query $\widehat{\mathcal{Q}}$ not in $\mathbb{C}_{\mathcal{DB}}$ can be an arbitrary element of $\mathbb{C}_{\mathcal{DB}}$.*

*Proof (Proof (sketch)).* Since the query $\widehat{\mathcal{Q}}$ is domain independent, its answer does not change by varying the domain, so we pick the smallest of such domains, which is unique and coincides with $\mathbb{C}_{\mathcal{DB}}$. The additional constants in the query not in $\mathbb{C}_{\mathcal{DB}}$ can be interpreted in an arbitrary way, since the answer would not depend on the interpretation of these additional constants.

**Theorem 4 (Domain independent rewriting).** *If the* rewritten query $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ *(from Theorem 3) is domain independent, then:*
$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \textit{for every } \mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} =$
$\{\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}} \mid \mathcal{I}^{(\mathbb{C}_{\mathcal{DB}})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}}]}\}$.

*Proof (Proof (sketch)).* From Theorem 3 and Lemma 2.

*Example 5 (continues Example 4).* Given the database predicates `MEETING` and `ACTIVITY`, the constraints, and the database introduced in Example 1, the query $\texttt{Project}(x)$ is implicitly definable from the database predicates `MEETING` and `ACTIVITY`, it has the *explicit definition* $\forall x.\texttt{Project}(x) \leftrightarrow \texttt{ACTIVITY}(x) \wedge \neg\texttt{MEETING}(x)$, and therefore it can be rewritten as the *domain independent* query $\texttt{ACTIVITY}(x) \wedge \neg\texttt{MEETING}(x)$. Indeed, the rewritten query has the same answer $\{x \mapsto \texttt{p}\}$ found in Example 1. □

We describe now two relevant cases that help in deciding whether a rewritten query is domain independent under constraints.

**Theorem 5 (Domain independence from $\mathcal{KB}$).** *Given a domain independent query $\mathcal{Q}$ and a set of domain independent constraints $\mathcal{KB}$, the rewritten query $\widehat{\mathcal{Q}}$ (from Theorem 3) is domain independent.*

*Proof (Proof (sketch)).* Notation: we extend the meaning of $\cdot^{\mathcal{I}}$ to arbitrary (open or closed) formulas as usual.

- Since $\widehat{\mathcal{Q}}$ is the explicit definition of $\mathcal{Q}$: for every $\mathcal{I} \in \mathsf{M}(\mathcal{KB}) : \mathcal{Q}^{\mathcal{I}} = \widehat{\mathcal{Q}}^{\mathcal{I}}$.
- Since $\mathcal{KB}$ is domain independent:
  for every $\mathcal{I}, \mathcal{J}$ with $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}} : \mathcal{I} \in \mathsf{M}(\mathcal{KB})$ iff $\mathcal{J} \in \mathsf{M}(\mathcal{KB})$.
- Since $\mathcal{Q}$ is domain independent: for every $\mathcal{I}, \mathcal{J}$ with $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}} : \mathcal{Q}^{\mathcal{I}} = \widehat{\mathcal{Q}}^{\mathcal{J}}$.
- From the previous steps: for every $\mathcal{I}, \mathcal{J} \in \mathsf{M}(\mathcal{KB})$ with $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}} : \widehat{\mathcal{Q}}^{\mathcal{I}} = \widehat{\mathcal{Q}}^{\mathcal{J}}$.
- Since $\widehat{\mathcal{Q}}$ mentions only database predicates:
  for every $\mathcal{I}, \mathcal{J} \in \mathsf{M}(\mathcal{KB})$ with $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}} : \widehat{\mathcal{Q}}^{\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}}}} = \widehat{\mathcal{Q}}^{\mathcal{J}|_{\mathbb{P}_{\mathcal{DB}}}}$.
- Since $\mathcal{I}|_{\mathbb{P}_{\mathcal{DB}}}$ and $\mathcal{J}|_{\mathbb{P}_{\mathcal{DB}}}$ do not depend on $\mathcal{KB}$: for every $\mathcal{I}, \mathcal{J}$ with $\cdot^{\mathcal{I}} = \cdot^{\mathcal{J}} : \widehat{\mathcal{Q}}^{\mathcal{I}} = \widehat{\mathcal{Q}}^{\mathcal{J}}$.
- Therefore $\widehat{\mathcal{Q}}$ is domain independent.

This theorem is important since it shows that if we have a set of domain independent constraints and a domain independent query, then the rewritten query is domain independent as well, and so we can reduce the query answering under constraints into standard query answering of a database using standard SQL query evaluation technologies. It is worth noting that the most important typical constraints considered in the database literature are domain independent: *tuple generating dependencies* (thus including foreign keys) and *equality generating dependencies* (thus including primary keys) are domain independent.

**Lemma 3 (Domain independence of database constraints).** *Let* tuple generating dependencies *(TGDs) be closed formulas of the form* $(\forall \mathbb{X}.A_1 \wedge \ldots \wedge A_n \rightarrow \exists \mathbb{Y}.B_1 \wedge \ldots \wedge B_m)$*, and* equality generating dependencies *(EGDs) be closed formulas of the form* $(\forall \mathbb{X}.A_1 \wedge \ldots \wedge A_n \rightarrow \exists \mathbb{Y}.E)$ *– with $A_i$ and $B_j$ atomic formulas (excluding equality atoms) and $E$ an equality atom. TGDs and EGDs are domain independent formulas.*

*Proof (Proof (sketch)).* It is enough to show that whenever a TGD or a EGD is true in the fixed interpretation $\mathcal{I}_{(\mathcal{DB})}^{(\mathbb{C}_{\mathcal{DB}})}$ then it is true also in any interpretation $\mathcal{I}_{(\mathcal{DB})}$ with an arbitrary domain.

This implies that an implicitly definable domain independent query under tuple and equality generating dependencies can be rewritten into a domain independent query over the database predicates only.

**Theorem 6 (Ground domain independence).** *If $\mathbb{C}$ is a finite set, let's extend the database $\mathcal{DB}$ with a new unary relation $\top_{\mathbb{C}}$ containing all the constants in $\mathbb{C}$. If the* rewritten query $\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ *(from Theorem 3) is a ground domain independent query, then the original query answering problem in Definition 1 is equivalent to the query answering problem with a* range restricted *domain independent query:*

$$\{\Theta_{\mathbb{X}}^{\mathbb{C}} \mid \textit{for every } \mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}\} =$$
$$\{\Theta_{\mathbb{X}}^{\mathbb{C}_{\mathcal{DB}}} \mid \mathcal{I}_{(\mathcal{DB})}^{(\mathbb{C}_{\mathcal{DB}})}|_{\mathbb{P}_{\mathcal{DB}}} \models (\widehat{\mathcal{Q}}_{[\mathbb{X}]} \wedge \top_{\mathbb{C}}(x_1) \wedge \cdots \wedge \top_{\mathbb{C}}(x_n))_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}_{\mathcal{DB}}}]}\}.$$

*Proof (Proof (sketch)).* Since $\mathbb{C}_{\mathcal{DB}} = \mathbb{C}$, by Theorem 3 we get:

$\{\Theta^{\mathbb{C}}_{\mathbb{X}} \mid \text{for every } \mathcal{I}_{(\mathcal{DB})} \in \mathsf{M}(\mathcal{KB}) : \mathcal{I}_{(\mathcal{DB})} \models \mathcal{Q}_{[\mathbb{X}/\Theta^{\mathbb{C}}_{\mathbb{X}}]}\} =$

$\{\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}} \mid \text{for every } \Delta^{\mathcal{I}} \supseteq \mathbb{C}_{\mathcal{DB}} : \mathcal{I}^{(\Delta^{\mathcal{I}})}_{(\mathcal{DB})}|_{\mathbb{P}_{\mathcal{DB}}} \models \widehat{\mathcal{Q}}_{[\mathbb{X}/\Theta^{\mathbb{C}_{\mathcal{DB}}}_{\mathbb{X}}]}\}$.

Since the substitutions of the variables in $\widehat{\mathcal{Q}}$ range only over the active domain $\mathbb{C}_{\mathcal{DB}}$, we can add to the query the restriction on the free variables without affecting its meaning. But now the range restricted query is domain independent, and we can apply Lemma 2.

In Section 4.2 an application of this theorem to ontologies is shown.

To sum up, we have seen in this section that in order to compute the rewriting of a query $\mathcal{Q}$ to a database $\mathcal{DB}$ under constraints $\mathcal{KB}$, we need to:

1. verify the properties which guarantee a domain independent rewriting;
2. check the consistency of the database $\mathcal{DB}$ wrt the constraints $\mathcal{KB}$;
3. check that the query is implicitly definable from the database predicates $\mathbb{P}_{\mathcal{DB}}$, by checking $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall\mathbb{X}.\mathcal{Q}_{[\mathbb{X}]} \leftrightarrow \widetilde{\mathcal{Q}_{[\mathbb{X}]}}$;
4. compute the interpolant $\widehat{\mathcal{Q}}$ (Theorem 2) which will be the rewriting of $\mathcal{Q}$;
5. execute the domain independent rewriting $\widehat{\mathcal{Q}}$ over the database $\mathcal{DB}$ using standard SQL.

Step 3 is the critical one, and ideally we would like to study interesting decidable fragments of $\mathcal{FOL}$ which guarantee the termination of the step. On the other hand, step 4 is always guaranteed to terminate. Step 5 tells us that the data complexity of the query answering problem in this framework is in $\mathsf{AC}^0$.

# 4 Applying the framework

## 4.1 Conjunctive queries and views

In this section we instantiate the general framework presented so far to *conjunctive queries*. We show how the essential definitions specialise to this case, so to relate it to the known literature.

**Definition 7 (Conjunctive View Rewriting Problem).** *Let $Q$ be a conjunctive query formulated over a fixed relational signature $\sigma$, $Q_i$ be conjunctive queries also formulated over $\sigma$, and $V_i$ fresh predicate symbols of arity equal to the number of free variables of the queries $Q_i$, $0 < i \leq k$. We say that $Q$* has a rewriting in terms of view definitions

$\mathcal{V} = \{\forall\mathbb{X}_i.V_i(\mathbb{X}_i) \leftrightarrow Q_i\}$

*if there is a (first-order) query $Q_{\mathcal{V}}$ over the signature $\{V_1, \ldots, V_k\}$ such that*

$\{\forall\mathbb{X}_i.V_i(\mathbb{X}_i) \leftrightarrow Q_i\} \models \forall\mathbb{X}.Q \leftrightarrow Q_{\mathcal{V}}$,

*where $\mathbb{X}_i$ and $\mathbb{X}$ are free variables of $Q_i$ and $Q$, respectively.*

The set $\{V_1, \ldots, V_k\}$ serves the role of the *database predicates*. For simplicity we use only binary relations in $\sigma$ as it is easy to see that, in the case of conjunctive queries and views, higher arity relations can be dealt with using reification.

Also, in the following, we allow only *connected* conjunctive views: views in which every quantified variable is connected to an answer variable through a sequence of relations.

*Example 6 ([13]).* Let $Q(x,y) \leftrightarrow \exists z,v,u.R(z,x),R(z,v),R(v,u),R(u,y)$, and
$V_1(x,y) \leftrightarrow \exists z,v.R(z,x),R(z,v),R(v,y)$,
$V_2(x,y) \leftrightarrow \exists z.R(x,z),R(z,y)$, and
$V_3(x,y) \leftrightarrow \exists z,v.R(x,z),R(z,v),R(v,y)$
(As usual we omit the leading universal quantifiers in the view definitions.) It has been shown that $Q$ has a rewriting in terms of $\{V_1,V_2,V_3\}$ of the form $\exists z.V_1(x,z) \wedge \forall v.(V_2(v,z) \rightarrow V_3(v,y))$.

Moreover, Nash *et. al.* [13] have shown that in the above case, a *conjunctive* rewriting cannot exist and have constructed a first-order rewriting. We show that such rewritings can be obtained effectively.

**Definition 8.** *Let $Q$ be a conjunctive query and $\mathcal{V} = \{\forall \mathbb{X}_i.V_i(\mathbb{X}_i) \leftrightarrow Q_i\}$ a set of conjunctive views over $\{R_1,\ldots,R_n\}$. We define a theory*
$\Sigma_{\mathcal{V}} = \{\forall \mathbb{X}_i.Q_i \rightarrow \widetilde{Q}_i, \forall \mathbb{X}_i.Q_i \leftarrow \widetilde{Q}_i \mid 0 < i \leq k\}$
*where $\widetilde{Q}_i$ is obtained from $Q_i$ by substituting a symbol $\widetilde{R}_j$ for $R_j$ for all $0 < i \leq k$ and $0 < j \leq n$. We call the formulas $\forall \mathbb{X}_i.Q_i \rightarrow \widetilde{Q}_i$ and $\forall \mathbb{X}_i.Q_i \leftarrow \widetilde{Q}_i$* rules *(associated with a view $V_i$; note that there are* two *rules associated with each view: one from left to right and one from right to left).*

Explicit definability of $Q$ in terms of $\mathcal{V}$ is defined as follows (see Theorem 1):

**Proposition 1 (Explicit Definability for Conjunctive Views).** *Let $Q$ be a conjunctive query and $\mathcal{V}$ a set of conjunctive views over $\{R_1,\ldots,R_n\}$. Then $Q$ has a rewriting in terms of $\mathcal{V}$ if and only if*
$\Sigma_{\mathcal{V}} \models \forall \mathbb{X}.Q \leftrightarrow \widetilde{Q}$,
*where $\widetilde{Q}$ is obtained from $Q$ by substituting a symbol $\widetilde{R}_i$ for $R_i$.*

There are several steps for this to be practical:

1. We need a decision procedure for $\Sigma_{\mathcal{V}} \models \forall \mathbb{X}.Q \leftrightarrow \widetilde{Q}$ to determine the existence of $Q_{\mathcal{V}}$,
2. We must be able to find $Q_{\mathcal{V}}$ effectively, and
3. We must make certain that $Q_{\mathcal{V}}$ is domain independent.

The last two issues have been addressed by the general framework, as conjunctive queries and views are domain independent; in the technical report [6] we did focus on the first issue:

**Theorem 7.** *Let $Q$ be a conjunctive query and $\mathcal{V}$ a set of conjunctive views over $\{R_1,\ldots,R_n\}$ such that a rewriting over $\mathcal{V}$ w.r.t. the views exists. Then there is an algorithm that, given $Q$ and $\mathcal{V}$, computes $Q_{\mathcal{V}}$.*

### 4.2 Guarded fragment and ontology languages

In this section, we briefly introduce the specialisation of the framework presented in Section 3 in the case of constraints expressed as Description logics (DLs) axioms [4]. The main motivation of this work is to address the scalability of query answering over large data sets in DL knowledge bases. This is of particular interest because the data complexity of the query answering problem in expressive

DLs is intractable whereas our framework allows one to use standard SQL query evaluation techniques.

Data in a DL knowledge base is typically stored in the so-called ABox component. An ABox differs from a database since it has an *open-world* semantics as in sound views and it is incompatible with the database (exact views) semantics as defined in Section 2. There are approaches to deal with ABoxes using standard relational database technology: for example, in the DL-Lite DL the expressivity of the description logic language is restricted in order to reduce query answering to SQL querying (see, e.g., [2]). But differently from DL-Lite, in our work we want to focus on knowledge bases with a database (exact views) semantics for the data – called DBox.

In [14], the problem of query answering under $\mathcal{ALC}$ DL constraints is studied using the DBox semantics as defined in Section 2. Here we generalise the approach by expressing the constraints in the decidable *guarded fragment* $\mathcal{GF}$ of $\mathcal{FOL}$ (already considered by [12]), which includes $\mathcal{ALC}$ and most DL languages without counting quantifiers. We prove now a general theorem which guarantees that the rewriting under $\mathcal{GF}$ constraints of an implicitly definable query is always a ground domain independent query in $\mathcal{GF}$ (due the Beth definability property for $\mathcal{GF}$ [11]), and so we can always use SQL to evaluate the rewritten query. In the technical report [6], we have proved the following:

**Theorem 8 (Ground domain independence of $\mathcal{GF}$).** *Queries in the guarded fragment $\mathcal{GF}$ of $\mathcal{FOL}$ are ground domain independent.*

## 5 Conclusions and future work

The paper presented a general framework for finding equivalent query rewritings in the context of combined knowledge and data bases, which used standard theorem proving techniques to find them. The question of when the rewritings can be efficiently evaluated was addressed, and two applications of the framework were shown: one dealing with an ontology language, and the other with a useful subclass of conjunctive views (which was generally an open problem). More details can be found in the technical report [6].

In addition to defining the general framework and to the theoretical contributions of the paper, we have also started experimental evaluation of the proposed techniques with the help of a state of the art first-order theorem prover. Indeed, the resolution proof for the running example (Example 6) is generated automatically in a fraction of a second.

We have begun work on several topics:

- Practical approaches to query evaluation face additional complications, such as dealing with binding patterns, duplicate semantics, ordering of data, size of the rewritings, and costs of executing relational operators, all of which impact on performance of query engines. Of particular interest is extracting *alternative* query reformulations from proofs of explicit definability.
- Many classes of constraints for which reasoning is decidable have been proposed both for databases and in the area of knowledge representation. However, constraints have to be balanced against the kinds of queries that can

be expressed. We are investigating various combinations (such as description logics combined with positive first-order queries) in our framework.

Further, we plan to extend our decidability result to the class of all (i.e., including disconnected) conjunctive views based on a decision procedure for testing whether an explicit definition exists.

## References


1. Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proc. PODS*, pages 254–263, 1998.
2. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-lite family and relations. *J. Artif. Int. Res.*, 36(1):1–69, 2009.
3. Arnon Avron. Constructibility and decidability versus domain independence and absoluteness. *Theor. Comput. Sci.*, 394(3):144–158, 2008.
4. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
5. E. W. Beth. On Padoa's methods in the theory of definitions. *Koninklijke Nederlandse Akademie van Wetenschappen, Proceedings*, 56:330–339, 1953. also Indagationes mathematicae, vol. 15.
6. Alex Borgida, Jos de Bruijn, Enrico Franconi, Inanç Seylan, Umberto Straccia, David Toman, and Grant Weddell. On finding query rewritings under expressive constraints. Technical report, Free University of Bozen-Bolzano, 2010. `http://www.inf.unibz.it/~franconi/papers/`.
7. William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.
8. Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.
9. Oren Etzioni, Keith Golden, and Daniel S. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89(1–2):113–148, 1997.
10. Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.
11. Eva Hoogland, Marten Marx, and Martin Otto. Beth definability for the guarded fragment. In *Proceedings of LPAR'99*, volume 1705 of *LNAI*, pages 273–285. Springer-Verlag, 1999.
12. Maarten Marx. Queries determined by views: pack your views. In *Proc. PODS*, pages 23–30, 2007.
13. Alan Nash, Luc Segoufin, and Victor Vianu. Determinacy and rewriting of conjunctive queries using views: A progress report. In *Proc. ICDT*, pages 59–73, 2007.
14. İnanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over DBoxes. In Craig Boutilier, editor, *IJCAI*, pages 923–925, 2009.
15. Alfred Tarski. Some methodological investigations on the definability of concepts. In *Logic, Semantics and Metamathematics*, pages 296–319. Clarendon Press, Oxford, UK, 1956.