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] Grant Weddell[4]

16th December 2009 15:31:31 +0100, by efranconi, SVN 862

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 novel results of applying this framework include: 1) if the original constraints are domain independent, then so will be the query rewritten in terms of database predicates; 2) for infinite databases, the rewriting of conjunctive queries over connected views is decidable; 3) one can apply this technique to the guarded fragment of FOL, obtaining results about ontology languages.

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.* [16]: Is it decidable whether there is a rewriting of a conjunctive query over connected conjunctive views?
- We show 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 [16].
- We also show an instantiation of the framework for constraints expressed in the guarded fragment of first-order logic (which can be seen as 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 and 5 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.

[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

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 φ, 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 φ_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 ϵ 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 φ, the (possibly empty) set $\mathbb{X}$ of all free variables of φ, a substitution $\Theta^{\Delta}_{\mathbb{X}}$, we say that $\mathcal{I}, \Theta^{\Delta}_{\mathbb{X}} \models \varphi$ iff φ 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 φ 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., [10]), *exact views* in databases (see, e.g., [1, 9, 16]), or *DBox* in KR (see [17]).

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 [4].

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* $\Theta^{\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}}_{\mathbb{X}}$*:*

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

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 see in the next Section 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 5) 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 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* $\texttt{MEETING}$ *and* $\texttt{ACTIVITY}$*, 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 [15, 16].

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

An alternative *equivalent* reformulation of the *semantic* definition of implicit definability is the following *syntactic* definition [18].

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 5, since in the guarded fragment of $\mathcal{FOL}$ unrestricted models entailment and finite models entailment coincide [12]. 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 [6] and Craig [8] 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, 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 [8] gave to this theorem a constructive proof, i.e., a procedure for computing explicit definitions. The constructive proof of Craig relies on the notion of *interpolation*. We review a procedure for interpolant calculation based on Tableau proofs in Appendix A.

Lemma 2 (Craig's interpolation)**.** *If $\varphi \rightarrow \psi$ is a valid formula in $\mathcal{FOL}$ and neither φ nor ψ are valid, then there is a formula χ in $\mathcal{FOL}$, called* interpolant*, whose predicate and constant symbols are among the predicate and constant symbols of both φ and ψ, 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 3 (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 can now combine these results to reduce answering of a definable query to a database under constraints as in Definition 1 to answering of a rewritten query to the database only.

Theorem 4 (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 3) in $\mathcal{FOL}$ with $\sigma(\widehat{\mathcal{Q}}) \subseteq \mathbb{P}_{\mathcal{DB}}$ such that:*

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

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 5 (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 (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 6 (Domain independent rewriting)**.** *If the* rewritten query *$\widehat{\mathcal{Q}}_{[\mathbb{X}]}$ (from Theorem 4) 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 (sketch). From Theorem 4 and Lemma 5. □

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 7 (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 4) is domain independent.*

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 very important since it shows that if we have a set of domain independent constraints and a domain independent query, we are guaranteed that 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 8 (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 (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 9 (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 4) 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 (sketch). Since $\mathbb{C}_{\mathcal{DB}} = \mathbb{C}$, by Theorem 4 we get:

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

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 5. □

We will see in Section 5 an important application of this theorem to description logics constraints.

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 $\widehat{\mathcal{Q}}$ according to Theorem 3, 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 AC^0. In the appendix B our running example is fully worked out along the steps specified above.

4 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, as this will lead us to the main insight needed to developing the decision procedure for finding rewritings in this setting.

Definition 7 (Conjunctive View Rewriting Problem)**.** *Let Q be a conjunctive query formulated over a fixed relational signature σ, Q_i be conjunctive queries also formulated over σ, 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 σ 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 ([16])**.** *Let $Q(x, y) \leftrightarrow \exists z, v, u.R(z, x), R(z, v), R(v, u), R(u, y)$, and*

$$\begin{array}{l} V_1(x, y) \leftrightarrow \exists z, v.R(z, x), R(z, v), R(v, y), \\ V_2(x, y) \leftrightarrow \exists z.R(x, z), R(z, y), \text{and} \\ V_3(x, y) \leftrightarrow \exists z, v.R(x, z), R(z, v), R(v, y) \end{array}$$

(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.* [16] 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 $\mathcal{V}$ can be characterised as follows (see Theorem 1):

Proposition 10 (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 already been addressed by the general framework, as conjunctive queries and views, when regarded to be constraints, are domain independent; in the sequel we thus concentrate on the first issue and only briefly discuss effectiveness of finding the rewriting using one of the known effective algorithms for construction of interpolants from proofs of interpolant existence [11, 14].

4.1 Decidability of Rewriting Existence for Conjunctive Views

We use a modified version of chase [2] for this purpose. Recall that we only use binary relations in the conjunctive queries; views, however, can be of any arity. In the following we can therefore assume that the chase corresponds to a labelled directed graph G whose nodes are numbered by integers and whose directed edges are labelled by a relation name.

Definition 9 (Level Based Complete Chase)**.** *Let Q be a conjunctive query, $\mathcal{V}$ a set of conjunctive views over $\{R_1, \ldots, R_n\}$. We define $\mathsf{apply}^{lr}_{\mathcal{V}}(G)$ to be an application of a "left-to-right" rule associated with the view V to the (partial) chase G in the standard fashion TGDs apply in a chase; in addition we disallow creating unnecessary anonymous objects. $\mathsf{apply}^{rl}_{\mathcal{V}}(C)$ is defined symmetrically and, more generally, both $\mathsf{apply}^{lr}_{\mathcal{V}}(C)$ and $\mathsf{apply}^{rl}_{\mathcal{V}}(C)$ are defined as the exhaustive application of all left-to-right and right-to-left rules for all $V \in \mathcal{V}$ on C, respectively. The integer substitutions for variables are called the* nodes of the chase*, and the* level based chase *is defined as follows:*

$$\begin{aligned} \mathsf{chase}^{0}_{\mathcal{V}}(Q) &= \{R_l(i,j) | R_l(x_i, x_j) \in Q\}, \\ \mathsf{chase}^{2k}_{\mathcal{V}}(Q) &= \mathsf{apply}^{rl}_{\mathcal{V}}(\mathsf{chase}^{2k-1}_{\mathcal{V}}(Q)) \textit{and} \\ \mathsf{chase}^{2k+1}_{\mathcal{V}}(Q) &= \mathsf{apply}^{lr}_{\mathcal{V}}(\mathsf{chase}^{2k}_{\mathcal{V}}(Q)). \end{aligned}$$

We call $\mathsf{chase}^{k}_{\mathcal{V}}(Q)$ the k-th round *of the chase and define the* level based complete chase *to be $\mathsf{chase}_{\mathcal{V}}(Q) = \bigcup_{k\geq 0} \mathsf{chase}^{k}_{\mathcal{V}}(Q)$.*

Observe that, due to the structure of constraints in $\Sigma_{\mathcal{V}}$, the chase is organised in two separate left-to-right and right-to-left rounds. Note that the heads and bodies of the rules switch their rôles in these two rounds. Since we disallow creating any new witnesses when they already exist, each of these phases terminates after finitely many applications of the rules: only symbols $\widetilde{R}_i$ (R_i, respectively) are generated in each round.

Proposition 11 ([2])**.** *$\Sigma_{\mathcal{V}} \models \forall \mathbb{X}. Q \to \widetilde{Q}$ if and only if $\widetilde{Q} \xrightarrow{\text{hom}} \mathsf{chase}_{\mathcal{V}}(Q)$, where $\xrightarrow{\text{hom}}$ is a homomorphism from the query that maps query variables of $\widetilde{Q}$ to the nodes corresponding to query variables of Q.*

Note that it is enough to test the implication $Q \to \widetilde{Q}$ rather than $Q \leftrightarrow \widetilde{Q}$ since the problem is symmetric.

Example 7. *The chase rounds for $Q(0,1)$ from Example 6:*

$$\begin{aligned} R(2,0), R(2,3), &R(3,4), R(4,1) \xrightarrow{V_1} \\ &\widetilde{R}(5,0), \widetilde{R}(5,6), \widetilde{R}(6,4) \xrightarrow{V_2} \\ R(2,0), R(2,3), &R(3,4), R(4,1), R(5,7), R(7,4) \xrightarrow{V_3} \\ &\widetilde{R}(5,0), \widetilde{R}(5,6), \widetilde{R}(6,4), \widetilde{R}(5,8), \widetilde{R}(8,9), \widetilde{R}(9,1) \end{aligned}$$

The last line contains a homomorphic image of $\widetilde{Q}(0,1)$ (and note that not all constraints were "applied" to keep the example simple).

In general, there is no guarantee that the chase terminates; indeed it is not difficult to construct views for which the chase is infinite. We develop a *blocking chase* incorporating a blocking condition for firing a rule that ensures that the chase terminates and for which Proposition 11 remains intact.

Our blocking condition depends critically on the *symmetry* of the constraints in $\Sigma_{\mathcal{V}}$. In particular, this symmetry ensures that any fresh nodes introduced by the (complete) chase rounds can trace their *parenthood* to other nodes already present in the (partial) chase and can therefore be organised in generations.

Definition 10 (Chase Predecessor)**.** *Let n be a node introduced by a rule in $\Sigma_{\mathcal{V}}$ for which n was a witness for a variable x in the head of the rule. Then there is a node m that was matched by the same variable x in the body of the rule. We say that m is an immediate predecessor of n.*

At every round of the chase we record the immediate predecessors for all nodes that appear in the chase (nodes inherited from the previous round by virtue of matching a universally quantified variable for some $\Sigma_{\mathcal{V}}$ are their own predecessors). In this way, for every tuple of constants appearing in the chase we can define a tuple of same generation *k-predecessors (for any k up to the number of chase rounds) by taking the k^{th} immediate predecessor for each of the nodes.*

This observation leads to the following Lemma:

Lemma 12. *If a rule body matches a state of the chase with a substitution θ then, for any number of chase rounds k, it also matches with a substitution θ' in which the constants are replaced by their k-predecessors.*

Proof (sketch). Consider a match consisting of constants, some of which were created by the last round of the chase. There is a homomorphism from these constants to their immediate predecessors due to symmetry of rules in $\Sigma_{\mathcal{V}}$ (each rule can be run *backwards*). Repeating this process and composing with the query match yields the result. □

This lemma allows us to prune spurious applications of the rules. It also allows us to determine which views are possibly relevant to a rewriting of a particular query: it is only those that match the (initial chase graph of the) original query.

Definition 11 (Neighborhood)**.** *Let Q be a conjunctive query, and $\mathcal{V}$ a set of connected conjunctive view definitions. We write $\mathsf{width}(Q, \mathcal{V})$ to denote the length of the longest simple path occurring in the body of Q or of some $V_i \in \mathcal{V}$. In addition, given the context of a non-negative integer k, we then write $\mathcal{L}_i$ to denote*

$$\{R(m,n) \mid R(m,n) \in \mathsf{chase}^k_{\mathcal{V}}(Q) \wedge i \rightsquigarrow m \wedge i \rightsquigarrow n\},$$

where $p \rightsquigarrow q$ holds when there exists an undirected simple path in $\mathsf{chase}^k_{\mathcal{V}}(Q)$ of length not exceeding $\mathsf{width}(Q, \mathcal{V})$.

Definition 12 (Level Based Blocking Chase)**.** *Let Q be a conjunctive query, $\mathcal{V}$ a set of connected conjunctive view definitions, and k a non-negative integer. For $G = \mathsf{chase}^k_{\mathcal{V}}(Q)$, let (V_i, S) denote the possibility for either of the two rules for view $V_i \in \mathcal{V}$ to match nodes $S \subseteq V_G$. We say that the application of the rule (V_i, S) is* blocked *in G if there exists $j > 1$ such that for each $p \in S$ we have a homomorphism $\mathcal{L}_p \xrightarrow{\text{hom}} \mathcal{L}_q$ where q is the j^{th} predecessor of node p.*

We define the level based blocking chase $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(C)$ *to be the level based complete chase in which, for all k, no left-to-right rule for V is fired on nodes S occurring in $G = \mathsf{chase}^k_{\mathcal{V}}(Q)$ for which (V, S) is blocked in G.*

Note that the homomorphism above requires nodes representing the distinguished variables of the original query to be mapped to themselves, in addition to preserving of all labelled edges. Also, it is sufficient to block the left-to-right rules as every individual round of the chase terminates and observe that all applications of those rules can trace their matches to the original query.

Lemma 13. *Let Q be a connected conjunctive query and $\mathcal{V}$ a set of connected conjunctive views over $\{R_1, \ldots, R_n\}$. Also let $\mathsf{chase}_{\mathcal{V}}(Q)$ and $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$ be level based complete and blocked chases of Q using $\Sigma_{\mathcal{V}}$, respectively. Then $\widetilde{Q} \xrightarrow{\text{hom}} \mathsf{chase}_{\mathcal{V}}(Q)$ if and only if $\widetilde{Q} \xrightarrow{\text{hom}} \mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$.*

Proof (sketch). The "if" direction is immediate. For the "only-if" direction, consider any sequence of rule applications of length k in $\mathsf{chase}_{\mathcal{V}}(Q)$ starting with a blocked rule. We show that the result of this sequence has a homomorphic image in $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$ by induction on k:

For $k = 1$, the claim follows immediately from the definition of blocking; for $k = i > 1$ we assume that the result of all sequences of length $i - 1$ has a homomorphic image in $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$. Then, however, the last rule matches this image and thus has been applied in $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$. Hence the result of its application in $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$ has a homomorphic image in $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q)$. □

Lemma 14. *For a given conjunctive query Q and a set of connected conjunctive view definitions $\mathcal{V}$, there exists k such that $\mathsf{chase}^{\text{blk}}_{\mathcal{V}}(Q) \subseteq \mathsf{chase}^k_{\mathcal{V}}(Q)$.*

Proof (sketch). It is straightforward to first show that there are a finite number of homomorphically distinct possibilities for $\mathcal{L}_i$ in $\mathsf{chase}_{\mathcal{V}}(C)$ by appeal to Lemma 12. It then follows, again straightforwardly, that there exists k such that, by Lemma 12, all possibilities (V, S) for firing a rule in $\mathsf{chase}^k_{\mathcal{V}}(Q)$ are blocked. □

Our main result for this section now follows from Proposition 11 together with Lemmas 13 and 14 above.

Theorem 15. *For a given conjunctive query Q and a set of connected conjunctive view definitions $\mathcal{V}$, the implication problem $\Sigma_{\mathcal{V}} \models \forall \mathbb{X}.Q \leftrightarrow \widetilde{Q}$ is decidable.*

4.2 Effective Construction of the Rewriting

We use Tableau or Resolution to construct a Craig interpolant for the first-order constraints representing the conjunctive views. Since we can determine whether the interpolant exists in advance using Theorem 15, the success of the construction is guaranteed in finite time.

Theorem 16. *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 $Q_{\mathcal{V}}$.*

Proof (sketch). We use a Tableau-based construction of Craig interpolants [11] as outlined in Appendix A; note that the tableau closes after finitely many steps since Q is known to be explicitly definable w.r.t. $\mathcal{V}$. □

A similar result can be shown for resolution-based proofs [14] obtaining an interpolant essentially identical to the hand-constructed rewriting presented in [16]. The corresponding resolution proof confirming the existence of the rewriting is presented in Appendix C.

5 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 [5]. 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., [3]). 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 [17], 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 [15]), which includes $\mathcal{ALC}$. 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 $\mathcal{GF}$ [13]), and so we can always use SQL to evaluate the rewritten query.

Theorem 17 (Ground domain independence of $\mathcal{GF}$)**.** *Queries in the guarded fragment $\mathcal{GF}$ of $\mathcal{FOL}$ are ground domain independent.*

Proof. See Appendix D □

6 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).

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 presented in Appendix C 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 crucially impact on performance of query engines. Of particular interest is extracting *alternative* query reformulations from first-order 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 have started to investigate 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] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. DL-Lite in the light of First-Order Logic. In *Proc. of the 22nd AAAI Conference on Artificial Intelligence*, pages 361–366, July 2007.

[4] Arnon Avron. Constructibility and decidability versus domain independence and absoluteness. *Theor. Comput. Sci.*, 394(3):144–158, 2008.

[5] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[6] 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.

[7] Patrick Blackburn and Maarten Marx. Constructive interpolation in hybrid logic. *Journal of Symbolic Logic*, 68(2):463–480, 2003.

[8] 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.

[9] Alin Deutsch, Lucian Popa, and Val Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[10] Oren Etzioni, Keith Golden, and Daniel S. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89(1–2):113–148, 1997.

[11] M. Fitting. *First-order logic and automated theorem proving (2nd ed.)*. Springer-Verlag, 1996.

[12] Erich Grädel. On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742, 1999.

[13] 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.

[14] Guoxiang Huang. Constructing Craig interpolation formulas. In *Computing and Combinatorics, LNCS 959*, pages 181–190, 1995.

[15] Maarten Marx. Queries determined by views: pack your views. In *Proc. PODS*, pages 23–30, 2007.

[16] 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.

[17] İnanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over DBoxes. In Craig Boutilier, editor, *IJCAI*, pages 923–925, 2009.

[18] Alfred Tarski. Some methodological investigations on the definability of concepts. In *Logic, Semantics and Metamathematics*, pages 296–319. Clarendon Press, Oxford, UK, 1956.

A Tableau Expansion and Interpolation Rules

Validity of a formula $\varphi \to \psi$ is checked by constructing a closed biased tableau whose root is labeled with $\{L(\varphi), R(\neg\psi)\}$. Interpolant calculation rules are then applied to the closed tableau, yielding an interpolant I of φ, ψ. We review the expansion rules and interpolant calculation rules by Fitting [11] and present the interpolant calculation rules for the case of FOL with equality. The biased expansion and interpolant calculation for the case of FOL with equality are analogous to the corresponding rules for constructing interpolants in quantified hybrid logic by Blackburn and Marx [7].

We first review the biased tableau expansion rules, which are straightforwardly obtained from the usual expansion rules. Here, X, Y are L or R. We start with the propositional rules

negation rules			α-rule	β-rule
$\dfrac{X(\neg\neg\varphi)}{X(\varphi)}$	$\dfrac{X(\neg\top)}{X(\bot)}$	$\dfrac{X(\neg\bot)}{X(\top)}$	$\dfrac{X(\varphi_1 \wedge \varphi_2)}{\begin{array}{c}X(\varphi_1)\\ X(\varphi_2)\end{array}}$	$\dfrac{X(\neg(\neg\varphi_1 \wedge \neg\varphi_2))}{X(\varphi_1) \mid X(\varphi_2)}$

Here, $\varphi, \varphi_1, \varphi_2$ are formulae.

We proceed with the first-order expansion rules. We write a formula φ with each free occurrence of the variable x replaced with term t as $\varphi(t)$. $\mathbb{C}^{par}$ extends $\mathbb{C}$ with an infinite set of new constants, called parameters. A parameter is *new* if it does not occur anywhere in the tableau.

γ-rule	δ-rule
$\dfrac{X(\forall x.\varphi)}{\begin{array}{c}X(\varphi(t))\\ (\text{for any } t \in \mathbb{C}^{par})\end{array}}$	$\dfrac{X(\exists x.\varphi)}{\begin{array}{c}X(\varphi(p))\\ (\text{for a new}\\ \text{parameter } p)\end{array}}$

Finally, we review the equality rules.

reflexivity	Tableau replacement
$\dfrac{X(\varphi)}{\begin{array}{c}X(t=t)\\ (t \in \mathbb{C}^{par}\\ \text{occurs in } \varphi)\end{array}}$	$\dfrac{\begin{array}{c}X(t=u)\\ Y(\varphi(t))\end{array}}{Y(\varphi(u))}$

Note that the biased reflexivity rule is slightly different from the usual reflexivity rule, because it is necessary to correct where constants come from, in order to ensure that the interpolant constructed later does not contain constants that do not appear in both the original formulas.

For the interpolant rules we use the notation $S \xrightarrow{int} I$, where $S = \{L(\varphi_1), \ldots, L(\varphi_n), R(\psi_1), \ldots, R(\psi_m)\}$ is a set of biased sentences, to mean that I is interpolant for the sentence $\varphi_1 \wedge \cdots \wedge \varphi_n \to (\neg\psi_1, \ldots, \vee\neg\psi_m)$. Observe that if $\{L(\varphi), R(\neg\psi)\} \xrightarrow{int} I$, I is an interpolant of $\varphi \to \psi$, as desired.

We start with the calculation rules for closed branches.

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

We proceed with the calculation rules corresponding to the respective tableau expansion rules. We start with the propositional case.

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

$$\frac{S \cup \{X(\varphi_1), X(\varphi_2)\} \xrightarrow{int} I}{S \cup \{X(\varphi_1 \wedge \varphi_2)\} \xrightarrow{int} I} \qquad \frac{S \cup \{L(\varphi_1)\} \xrightarrow{int} I_1 \quad S \cup \{L(\varphi_2)\} \xrightarrow{int} I_2}{S \cup \{L(\neg(\neg\varphi_1 \wedge \neg\varphi_2))\} \xrightarrow{int} I_1 \vee I_2}$$

$$\frac{S \cup \{R(\varphi_1)\} \xrightarrow{int} I_1 \quad S \cup \{R(\varphi_2)\} \xrightarrow{int} I_2}{S \cup \{R(\neg(\neg\varphi_1 \wedge \neg\varphi_2))\} \xrightarrow{int} I_1 \wedge I_2}$$

For the first-order case, assume $S = \{L(\phi_1), \ldots, L(\phi_n), R(\psi_1), \ldots, R(\psi_m)\}$ and p is a parameter that does not occur in S or ϕ. In addition, with $\varphi[t_1/t_2]$ we denote the formula obtained from φ by replacing every occurrence of the term t_1 with t_2.

$$\frac{S \cup \{X(\varphi(p))\} \xrightarrow{int} I}{S \cup \{X(\exists x.\varphi(x))\} \xrightarrow{int} I}$$

$$\frac{S \cup \{L(\varphi(c))\} \xrightarrow{int} I}{S \cup \{L(\forall x.\varphi(x))\} \xrightarrow{int} I} \text{ if } c \text{ occurs in } \{\phi_1, \ldots, \phi_n\} \qquad \frac{S \cup \{R(\varphi(c))\} \xrightarrow{int} I}{S \cup \{R(\forall x.\varphi(x))\} \xrightarrow{int} I} \text{ if } c \text{ occurs in } \{\psi_1, \ldots, \psi_n\}$$

$$\frac{S \cup \{L(\varphi(c))\} \xrightarrow{int} I}{S \cup \{L(\forall x.\varphi(x))\} \xrightarrow{int} \forall x.I[c/x]} \text{ otherwise} \qquad \frac{S \cup \{R(\varphi(c))\} \xrightarrow{int} I}{S \cup \{R(\forall x.\varphi(x))\} \xrightarrow{int} \exists x.I[c/x]} \text{ otherwise}$$

We conclude with the calculation rules for equality.

$$\frac{S \cup \{X(\varphi), X(t=t)\} \xrightarrow{int} I}{S \cup \{X(\varphi)\} \xrightarrow{int} I} \qquad \frac{S \cup \{X(t=u), X(\varphi(u))\} \xrightarrow{int} I}{S \cup \{X(t=u), X(\varphi(t))\} \xrightarrow{int} I}$$

$$\frac{S \cup \{R(t=u), L(\varphi(u))\} \xrightarrow{int} I}{S \cup \{R(t=u), L(\varphi(t))\} \xrightarrow{int} t=u \to I} \text{ if } u \text{ occurs in } \{\varphi(t), \phi_1, \ldots, \phi_n\} \qquad \frac{S \cup \{L(t=u), R(\varphi(u))\} \xrightarrow{int} I}{S \cup \{L(t=u), R(\varphi(t))\} \xrightarrow{int} t=u \wedge I} \text{ if } u \text{ occurs in } \{\varphi(t), \psi_1, \ldots, \psi_m\}$$

$$\frac{S \cup \{R(t=u), L(\varphi(u))\} \xrightarrow{int} I}{S \cup \{R(t=u), L(\varphi(t))\} \xrightarrow{int} I[u/t]} \text{ otherwise} \qquad \frac{S \cup \{L(t=u), R(\varphi(u))\} \xrightarrow{int} I}{S \cup \{L(t=u), R(\varphi(t))\} \xrightarrow{int} I[u/t]} \text{ otherwise}$$

Correctness of the propositional and standard first-order interpolant calculation rules was shown by Fitting [11]. Correctness of the rules for equality was shown by Blackburn and Marx [7].

B Fully worked out example for Section 3

Example 8 (Example 1 complete)**.** *For the sake of clearness, let us sum-up all the tasks for query rewriting in case of Example 1. So, the database predicates are*

$$\texttt{MEETING} \quad \textit{and} \quad \texttt{ACTIVITY}\,.$$

The database $\mathcal{DB}$ is

$$\texttt{ACTIVITY}: \langle \texttt{m} \rangle\ , \texttt{ACTIVITY}: \langle \texttt{p} \rangle\ ,\ \textit{and}\ \texttt{MEETING}: \langle \texttt{m} \rangle\ .$$

The set $\mathcal{KB}$ of constraints is:

$$\forall x.\texttt{Project}(x) \to \texttt{ACTIVITY}(x)\ , \tag{1}$$
$$\forall x.\texttt{MEETING}(x) \to \texttt{ACTIVITY}(x)\ , \tag{2}$$
$$\forall x.\texttt{ACTIVITY}(x) \to (\texttt{Project}(x) \vee \texttt{MEETING}(x))\ , \tag{3}$$
$$\forall x.\texttt{Project}(x) \to \neg\texttt{MEETING}(x)\ . \tag{4}$$

The query is:

$$\texttt{Q}(x) \leftrightarrow \texttt{Project}(x)\ .$$

Now,

1. *we check the consistency of the database $\mathcal{DB}$ wrt the constraints $\mathcal{KB}$. To this end, consider the set of constraints $\mathcal{KB}_{\mathcal{DB}}$*

$$\begin{array}{l} \forall x.\texttt{ACTIVITY}(x) \leftrightarrow (x = \texttt{m}) \vee (x = \texttt{p})\ , \\ \forall x.\texttt{MEETING}(x) \leftrightarrow (x = \texttt{m})\ , \\ (\texttt{m} \neq \texttt{p})\ . \end{array} \tag{5}$$

   We have that $\mathcal{DB}$ is consistent wrt the constraints $\mathcal{KB}$ iff $\mathcal{KB} \cup \mathcal{KB}_{\mathcal{DB}}$ has a model, which indeed can be verified using a FOL theorem prover.

2. *we check that the query is implicitly definable from the database predicates* $\mathbb{P}_{\mathcal{DB}}$. *To do so, we check (using a FOL theorem prover) whether* $\mathcal{KB} \cup \widetilde{\mathcal{KB}} \models \forall x.\mathtt{Q}(x) \leftrightarrow \widetilde{\mathtt{Q}}(x)$, *where*

$$\widetilde{\mathtt{Q}}(x) \leftrightarrow \widetilde{\mathtt{Project}}(x) \ ,$$

and $\widetilde{\mathcal{KB}}$ *is*

$$\forall x.\widetilde{\mathtt{Project}}(x) \rightarrow \mathtt{ACTIVITY}(x) \ , \tag{6}$$
$$\forall x.\mathtt{MEETING}(x) \rightarrow \mathtt{ACTIVITY}(x) \ , \tag{7}$$
$$\forall x.\mathtt{ACTIVITY}(x) \rightarrow (\widetilde{\mathtt{Project}}(x) \vee \mathtt{MEETING}(x)) \ , \tag{8}$$
$$\forall x.\widetilde{\mathtt{Project}}(x) \rightarrow \neg\mathtt{MEETING}(x) \ . \tag{9}$$

3. *we now compute the interpolant* $\widehat{\mathtt{Q}}(\mathtt{c})$ *according to Theorem 3. To this end, consider a new constant* $\mathtt{c}$. *We know by Theorem 3 that*

$$(\bigwedge \mathcal{KB} \wedge \mathtt{Q}(\mathtt{c})) \rightarrow (\bigwedge \widetilde{\mathcal{KB}} \rightarrow \widetilde{\mathtt{Q}}(\mathtt{c})) \tag{10}$$

is valid and, thus, we may compute an interpolant $\widehat{\mathtt{Q}}(\mathtt{c})$ *of it. Then* $\widehat{\mathtt{Q}}(x)$ *is the rewriting of* $\mathtt{Q}(x)$. *To show this step, let us consider the interpolation rules in Appendix A. Now, it is easily verified that the following holds:*

$$\frac{S \cup \{X(\varphi_1 \vee \varphi_2), X(\varphi_2)), X(\neg\varphi_1)\} \xrightarrow{int} I}{S \cup \{X(\varphi_1 \vee \varphi_2), X(\neg\varphi_1)\} \xrightarrow{int} I} \tag{11}$$

and, similarly,

$$S \cup \{R(\varphi_1 \vee \varphi_2), L(\neg\varphi_1), L(\neg\varphi_2)\} \xrightarrow{int} \varphi_1 \wedge \varphi_2 \ . \tag{12}$$

Now, as formula (10) *is valid, according to Appendix A, the interpolant computation starts with* $S_0 = \{L(\varphi), R(\neg\psi)\}$, *where* φ *is*

$$\bigwedge \mathcal{KB} \wedge \mathtt{Q}(\mathtt{c})$$

and $\neg\psi$ *is*

$$\bigwedge \widetilde{\mathcal{KB}} \wedge \neg\widetilde{\mathtt{Q}}(\mathtt{c}) \ .$$

As $\mathtt{c}$ *occurs in the labelled formulae, by applying the interpolation rule for* $\forall$, *we may instantiate/ground all universal quantified formulae in* $\mathcal{KB}$ *and* $\widetilde{\mathcal{KB}}$ *with constant* $\mathtt{c}$, *which we denote* $\mathcal{KB}_{[x/\mathtt{c}]}$ *and* $\widetilde{\mathcal{KB}}_{[x/\mathtt{c}]}$, *respectively. Now, by applying the interpolation rule for conjunction, it suffices to compute an interpolant for*

$$\mathbb{S} = \bigcup_{\varphi \in \mathcal{KB}_{[x/\mathtt{c}]}} \{L(\varphi)\} \cup \{L(\mathtt{Q}(\mathtt{c}))\} \cup \bigcup_{\psi \in \widetilde{\mathcal{KB}}_{[x/\mathtt{c}]}} \{R(\psi)\} \cup \{R(\neg\widetilde{\mathtt{Q}}(\mathtt{c}))\} \ .$$

Therefore, we have:

(a) $L(\neg\mathtt{Project}(\mathtt{c}) \vee \mathtt{ACTIVITY}(\mathtt{c}))$ *[Input]*
(b) $L(\neg\mathtt{MEETING}(\mathtt{c}) \vee \mathtt{ACTIVITY}(\mathtt{c}))$ *[Input]*
(c) $L(\neg\mathtt{ACTIVITY}(\mathtt{c}) \vee \mathtt{Project}(\mathtt{c}) \vee \mathtt{MEETING}(\mathtt{c}))$ *[Input]*
(d) $L(\neg\mathtt{Project}(\mathtt{c}) \vee \neg\mathtt{MEETING}(\mathtt{c}))$ *[Input]*
(e) $L(\mathtt{Q}(\mathtt{c}))$ *[Input]*
(f) $R(\neg\widetilde{\mathtt{Project}}(\mathtt{c}) \vee \mathtt{ACTIVITY}(\mathtt{c}))$ *[Input]*
(g) $R(\neg\mathtt{MEETING}(\mathtt{c}) \vee \mathtt{ACTIVITY}(\mathtt{c}))$ *[Input]*
(h) $R(\neg\mathtt{ACTIVITY}(\mathtt{c}) \vee \widetilde{\mathtt{Project}}(\mathtt{c}) \vee \mathtt{MEETING}(\mathtt{c}))$ *[Input]*

(i) $R(\neg\widetilde{\texttt{Project}}(\texttt{c}) \vee \neg\texttt{MEETING}(\texttt{c}))$ *[Input]*

(j) $R(\widetilde{\texttt{Q}}(\texttt{c}))$ *[Input]*

(k) $L(\texttt{ACTIVITY}(\texttt{c}))$ *[(11):a,e]*

(l) $L(\neg\texttt{MEETING}(\texttt{c}))$ *[(11):d,e]*

(m) $R(\neg\texttt{ACTIVITY}(\texttt{c}) \vee \texttt{MEETING}(\texttt{c}))$ *[(11):h,j]*

(n) $\widehat{\texttt{Q}}(\texttt{c}) := \texttt{ACTIVITY}(\texttt{c}) \wedge \neg\texttt{MEETING}(\texttt{c})$, *[(12):k,l,m]*

Therefore, the rewriting of $\texttt{Q}$ *is*

$$\forall x.\widehat{\texttt{Q}}(x) \leftrightarrow (\texttt{ACTIVITY}(x) \wedge \neg\texttt{MEETING}(x)) \ .$$

4. *we conclude by executing* $\widehat{\texttt{Q}}$ *over the database* $\mathcal{DB}$ *using standard SQL query evaluation technologies:*

```
SELECT  ACTIVITY.0  FROM  ACTIVITY , MEETING
        WHERE  ACTIVITY.0  !=  MEETING.0
```

and we get the answer $\{x \mapsto \texttt{p}\}$. □

C Resolution Proof of Existence of Rewriting in Example 6

1. $Q(sk_0, sk_1)$ [Input]
2. $\neg\widetilde{Q}(sk_0, sk_1)$ [Input]
3. $V_2(X_0, X_1) \vee \neg\widetilde{R}(X_0, X_3) \vee \neg\widetilde{R}(X_3, X_1)$ [Input]
4. $R(sk_3(X_0, X_1), X_1) \vee \neg V_2(X_0, X_1)$ [Input]
5. $R(X_0, sk_3(X_0, X_1)) \vee \neg V_2(X_0, X_1)$ [Input]
6. $\widetilde{R}(sk_4(X_0, X_1), sk_5(X_0, X_1)) \vee \neg V_1(X_0, X_1)$ [Input]
7. $\widetilde{R}(sk_5(X_0, X_1), X_1) \vee \neg V_1(X_0, X_1)$ [Input]
8. $\widetilde{R}(sk_4(X_0, X_1), X_0) \vee \neg V_1(X_0, X_1)$ [Input]
9. $\widetilde{R}(sk_6(X_0, X_1), sk_7(X_0, X_1)) \vee \neg V_3(X_0, X_1)$ [Input]
10. $\widetilde{R}(sk_7(X_0, X_1), X_1) \vee \neg V_3(X_0, X_1)$ [Input]
11. $\widetilde{R}(X_0, sk_6(X_0, X_1)) \vee \neg V_3(X_0, X_1)$ [Input]
12. $V_3(X_0, X_1) \vee \neg R(X_0, X_4) \vee \neg R(X_5, X_1) \vee \neg R(X_4, X_5)$ [Input]
13. $V_1(X_0, X_1) \vee \neg R(X_4, X_0) \vee \neg R(X_5, X_1) \vee \neg R(X_4, X_5)$ [Input]
14. $\widetilde{Q}(X_0, X_1) \vee \neg\widetilde{R}(X_2, X_3) \vee \neg\widetilde{R}(X_2, X_0) \vee \neg\widetilde{R}(X_4, X_1) \vee \neg\widetilde{R}(X_3, X_4)$ [Input]
15. $R(sk_{13}(X_0, X_1), sk_{14}(X_0, X_1)) \vee \neg Q(X_0, X_1)$ [Input]
16. $R(sk_14(X_0, X_1), X_1) \vee \neg Q(X_0, X_1)$ [Input]
17. $R(sk_12(X_0, X_1), X_0) \vee \neg Q(X_0, X_1)$ [Input]
18. $R(sk_{12}(X_0, X_1), sk_{13}(X_0, X_1)) \vee \neg Q(X_0, X_1)$ [Input]
19. $R(sk_{13}(sk_0, sk_1), sk_{14}(sk_0, sk_1))$ [Res:1,15]
20. $R(sk_{12}(sk_0, sk_1), sk_0)$ [Res:1,17]
21. $R(sk_{12}(sk_0, sk_1), sk_{13}(sk_0, sk_1))$ [Res:1,18]
22. $\neg R(sk_{12}(sk_0, sk_1), X_1) \vee \neg R(sk_{13}(sk_0, sk_1), X_2) \vee V_1(X_1, X_2)$ [Res:13,21]

23. $\neg R(sk_{13}(sk_0, sk_1), X_1) \vee V_1(sk_0, X_1)$ [Res:20,22]

24. $V_1(sk_0, sk_{14}(sk_0, sk_1))$ [Res:19,23]

25. $\neg\widetilde{R}(X_0, X_1) \vee \neg\widetilde{R}(X_0, sk_0) \vee \neg\widetilde{R}(X_2, sk_1) \vee \neg\widetilde{R}(X_1, X_2)$ [Res:2,14]

26. $\neg\widetilde{R}(sk_7(X_1, X_2), sk_1) \vee \neg\widetilde{R}(X_3, sk_6(X_1, X_2))$
$\vee \neg V_3(X_1, X_2) \vee \neg\widetilde{R}(X_3, sk_0)$ [Res:9,25]

27. $\neg\widetilde{R}(X_1, sk_6(X_2, sk_1)) \vee \neg V_3(X_2, sk_1) \vee \neg\widetilde{R}(X_1, sk_0)$ [Res:10,26]

28. $\neg V_3(X_1, sk_1) \vee \neg\widetilde{R}(X_1, sk_0)$ [Res:11,27]

29. $\neg\widetilde{R}(X_1, sk_5(X_2, X_3)) \vee V_2(X_1, X_3) \vee \neg V_1(X_2, X_3)$ [Res:3,7]

30. $V_2(sk_4(X_1, X_2), X_2) \vee \neg V_1(X_1, X_2)$ [Res:6,29]

31. $R(sk_14(sk_0, sk_1), sk_1)$ [Res:1,16]

32. $\neg R(X_1, sk_3(X_2, X_3)) \vee \neg R(X_3, X_4) \vee \neg V_2(X_2, X_3) \vee V_3(X_1, X_4)$ [Res:4,12]

33. $\neg R(X_1, X_2) \vee \neg V_2(X_3, X_1) \vee V_3(X_3, X_2)$ [Res:5,32]

34. $\neg V_2(X_1, sk_{14}(sk_0, sk_1)) \vee V_3(X_1, sk_1)$ [Res:31,33]

35. $V_3(sk_4(X_1, sk_{14}(sk_0, sk_1)), sk_1) \vee \neg V_1(X_1, sk_{14}(sk_0, sk_1))$ [Res:30,34]

36. $\neg\widetilde{R}(sk_4(X_1, sk_{14}(sk_0, sk_1)), sk_0) \vee \neg V_1(X_1, sk_{14}(sk_0, sk_1))$ [Res:28,35]

37. $\square$ [Res:8,24,36 (w/forward subsumption)]

D The Guarded Fragment $\mathcal{GF}$

In this section, we consider the decidable *guarded fragment* $\mathcal{GF}$ of $\mathcal{FOL}$ (Section 2). Since $\mathcal{FOL}$ is function-free, the set of terms consists of constants and variables. The set of $\mathcal{GF}$-formulas is the smallest set such that

1. Every $R(t_i, \ldots, t_n)$, where R is an n-place relation symbol in $\mathbb{P}$ and $t_1, \ldots, t_n$ are terms, is an atomic $\mathcal{GF}$-formula.
2. If φ is a $\mathcal{GF}$-formula then $\neg\varphi \in \mathcal{GF}$.
3. If φ, ψ are $\mathcal{GF}$-formulas then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are $\mathcal{GF}$-formulas.
4. If G is an atomic $\mathcal{GF}$-formula, φ is a $\mathcal{GF}$-formula, and $\mathbb{X}$ is a finite, non-empty sequence of variables such that $\mathbb{X} \subseteq \mathsf{free}(\varphi) \subseteq \mathsf{free}(G)$ then

$$\exists\mathbb{X}.G \wedge \psi \quad \text{and} \quad \forall\mathbb{X}.G \rightarrow \psi$$

are $\mathcal{GF}$-formulas.

Here $\mathsf{free}(\varphi)$ means the set of free variables of φ. A *sentence* (also called a *closed formula*) of $\mathcal{GF}$ is a formula of $\mathcal{GF}$ with no free-variable occurrences. An atom G that relativizes a quantifier as in rule 4 is the *guard* of the quantifier.

The semantics of $\mathcal{GF}$ is given by *interpretations* as usual. An *assignment* in an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ is a mapping $\mathcal{A}$ from the set of variables to the set $\Delta^{\mathcal{I}}$. We denote the image of the variable x under an assignment $\mathcal{A}$ by $x^{\mathcal{A}}$. To each term t, we associate a value $t^{\mathcal{I},\mathcal{A}}$ in $\Delta^{\mathcal{I}}$ as follows:

1. For a constant symbol c, $c^{\mathcal{I},\mathcal{A}} = c^{\mathcal{I}}$.
2. For a variable x, $x^{\mathcal{I},\mathcal{A}} = x^{\mathcal{A}}$.

Two assignments $\mathcal{A}$ and $\mathcal{B}$ in the interpretation $\mathcal{I}$ *agree on a set of variables* $\mathbb{X}$ if and only if $x^{\mathcal{A}} = x^{\mathcal{B}}$ for every $x \in \mathbb{X}$. An assignment $\mathcal{B}$ in the interpretation $\mathcal{I}$ is an $\mathbb{X}$-*variant* of the assignment $\mathcal{A}$, provided $\mathcal{A}$ and $\mathcal{B}$ agree on every variable except possibly the ones $\mathbb{X}$.

The truth of a formula φ w.r.t. an interpretation $\mathcal{I}$ and assignment $\mathcal{A}$ in $\mathcal{I}$, written $\mathcal{I}, \mathcal{A} \models \varphi$, is inductively defined as follows.

1. $\mathcal{I}, \mathcal{A} \models R(t_1, \ldots, t_n)$ iff $\langle t_1^{\mathcal{I},\mathcal{A}}, \ldots, t_n^{\mathcal{I},\mathcal{A}} \rangle \in R^{\mathcal{I}}$.
2. $\mathcal{I}, \mathcal{A} \models \neg\varphi$ iff $\mathcal{I}, \mathcal{A} \not\models \neg\varphi$.
3. $\mathcal{I}, \mathcal{A} \models \varphi \wedge \psi$ iff $\mathcal{I}, \mathcal{A} \models \varphi$ and $\mathcal{I}, \mathcal{A} \models \psi$.
4. $\mathcal{I}, \mathcal{A} \models \varphi \vee \psi$ iff $\mathcal{I}, \mathcal{A} \models \varphi$ or $\mathcal{I}, \mathcal{A} \models \psi$.
5. $\mathcal{I}, \mathcal{A} \models \forall\mathbb{X}.G \rightarrow \varphi$ iff $\mathcal{I}, \mathcal{B} \models G$ implies $\mathcal{I}, \mathcal{B} \models \varphi$ for every assignment $\mathcal{B}$ in $\mathcal{I}$ that is an $\mathbb{X}$-variant of $\mathcal{A}$.
6. $\mathcal{I}, \mathcal{A} \models \exists\mathbb{X}.G \wedge \varphi$ iff $\mathcal{I}, \mathcal{B} \models G$ and $\mathcal{I}, \mathcal{B} \models \varphi$ for some assignment $\mathcal{B}$ in $\mathcal{I}$ that is an $\mathbb{X}$-variant of $\mathcal{A}$.

A formula φ is *true in an interpretation* $\mathcal{I}$, written $\mathcal{I} \models \varphi$, provided $\mathcal{I}, \mathcal{A} \models \varphi$ for all assignments $\mathcal{A}$ in $\mathcal{I}$.

We will be considering $\mathcal{GF}$-formulas in negation normal form. A $\mathcal{GF}$-formula φ is in *negation normal form* if and only if the negation sign appears only in front of atomic formulas in φ. A $\mathcal{GF}$-formula φ can be rewritten into an equivalent formula φ' in negation normal form in time linear in the size of φ.

Proof of Theorem 17. Let $\mathcal{I}$ and $\mathcal{J}$ are two interpretations that agree on the interpretation of the predicates and constants. The following claim will be useful in proving the lemma.

Claim 18. *For every $\mathcal{GF}$-formula φ in negation normal form, if $\mathcal{A}$ is an assignment in $\mathcal{I}$ and $\mathcal{B}$ is an assignment in $\mathcal{J}$ such that $\mathcal{A}$ and $\mathcal{B}$ agree on $\mathsf{free}(\varphi)$ then $\mathcal{I}, \mathcal{A} \models \varphi$ if and only if $\mathcal{J}, \mathcal{B} \models \varphi$.*

Proof. We proceed by induction on the structure of φ which is in negation normal form.

1. $\varphi = R(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms. $\mathcal{I}, \mathcal{A} \models R(t_1, \ldots, t_n)$ iff $\langle t_1^{\mathcal{I},\mathcal{A}}, \ldots, t_n^{\mathcal{I},\mathcal{A}} \rangle \in R^{\mathcal{I}}$. Since $R^{\mathcal{I}} = R^{\mathcal{J}}$ and $\mathcal{A}$, $\mathcal{B}$ agree on $\mathsf{free}(\varphi)$, $\langle t_1^{\mathcal{I},\mathcal{B}}, \ldots, t_n^{\mathcal{I},\mathcal{B}} \rangle \in R^{\mathcal{J}}$, i.e., $\mathcal{J}, \mathcal{B} \models R(t_1, \ldots, t_n)$.
2. $\varphi = \neg R(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are terms. $\mathcal{I}, \mathcal{A} \models \neg R(t_1, \ldots, t_n)$ iff $\mathcal{I}, \mathcal{A} \not\models R(t_1, \ldots, t_n)$ iff $\langle t_1^{\mathcal{I},\mathcal{A}}, \ldots, t_n^{\mathcal{I},\mathcal{A}} \rangle \notin R^{\mathcal{I}}$. Since $R^{\mathcal{I}} = R^{\mathcal{J}}$ and $\mathcal{A}$, $\mathcal{B}$ agree on $\mathsf{free}(\varphi)$, $\langle t_1^{\mathcal{I},\mathcal{B}}, \ldots, t_n^{\mathcal{I},\mathcal{B}} \rangle \notin R^{\mathcal{J}}$, i.e., $\mathcal{J}, \mathcal{B} \models \neg R(t_1, \ldots, t_n)$.
3. $\varphi = \varphi_1 \wedge \varphi_2$. $\mathcal{I}, \mathcal{A} \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{I}, \mathcal{A} \models \varphi_1$ and $\mathcal{I}, \mathcal{A} \models \varphi_2$ iff by the inductive hypothesis, $\mathcal{J}, \mathcal{B} \models \varphi_1$ and $\mathcal{J}, \mathcal{B} \models \varphi_2$ iff $\mathcal{J}, \mathcal{B} \models \varphi_1 \wedge \varphi_2$.
4. $\varphi = \varphi_1 \vee \varphi_2$. Analogous to the previous case.
5. $\varphi = \exists\mathbb{Y}.G \wedge \psi$. $\mathcal{I}, \mathcal{A} \models \exists\mathbb{Y}.G \wedge \psi$ iff $\mathcal{I}, \mathcal{A}' \models G$ and $\mathcal{I}, \mathcal{A}' \models \psi$ for some assignment $\mathcal{A}'$ in $\mathcal{I}$ that is an $\mathbb{Y}$-variant of $\mathcal{A}$. Let $\mathcal{B}'$ be exactly like $\mathcal{B}$ except that $y^{\mathcal{B}'} = y^{\mathcal{A}'}$ for every $y \in \mathbb{Y}$. $\mathcal{B}'$ is an assignment in $\mathcal{J}$ because by $G^{\mathcal{I}} = G^{\mathcal{J}}$, $y^{\mathcal{B}'} = y^{\mathcal{A}'} \in \Delta^{\mathcal{J}}$, for every $y \in \mathbb{Y}$. Furthermore, $\mathcal{B}'$ is an $\mathbb{Y}$-variant of $\mathcal{B}$ by its construction. Since $\mathcal{A}'$ and $\mathcal{B}'$ agree on free variables in G, and $G^{\mathcal{I}} = G^{\mathcal{J}}$, we have $\mathcal{J}, \mathcal{B}' \models G$. We know that $\mathsf{free}(\psi) \subseteq \mathsf{free}(G)$, and since $\mathcal{A}'$ and $\mathcal{B}'$ agree on $\mathsf{free}(G)$, we have that $\mathcal{A}'$ and $\mathcal{B}'$ also agree on $\mathsf{free}(\psi)$. But then $\mathcal{J}, \mathcal{B}' \models \psi$ by the inductive hypothesis. Hence, $\mathcal{J}, \mathcal{B} \models \exists\mathbb{Y}.G \wedge \psi$.
6. $\varphi = \forall\mathbb{Y}.G \rightarrow \psi$. Analogous to the previous case.

□

Let $\mathcal{Q}_{[\mathbb{X}]}$ be a query in $\mathcal{GF}$. We need to show that $\mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$ is domain independent for every substitution $\Theta_{\mathbb{X}}^{\mathbb{C}}$. To this aim, we pick an arbitrary substitution $\Theta_{\mathbb{X}}^{\mathbb{C}}$ because then the desired result will follow by our choice of $\Theta_{\mathbb{X}}^{\mathbb{C}}$ being arbitrary. Suppose $\mathcal{A}$ and $\mathcal{B}$ are assignments in $\mathcal{I}$ and $\mathcal{J}$, respectively. Since $\mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$ is a sentence, $\mathcal{A}$ and $\mathcal{B}$ trivially agree on the free variables of $\mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$. But then by Claim 18, we have that $\mathcal{I}, \mathcal{A} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$ if and only if $\mathcal{J}, \mathcal{B} \models \mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$. Since this holds for any such assignment $\mathcal{A}$ and $\mathcal{B}$, we can conclude that $\mathcal{Q}_{[\mathbb{X}/\Theta_{\mathbb{X}}^{\mathbb{C}}]}$ is domain independent. □