Query Containment Using Views (extended abstract)*

Diego Calvanese[1], Giuseppe De Giacomo[1], Maurizio Lenzerini[1], Moshe Y. Vardi[2]

[1] Università di Roma "La Sapienza"
Dip. di Informatica e Sistemistica
via Salaria 113, 00198 Roma, Italy
lastname@dis.uniroma1.it

[2] Department of Computer Science
Rice University, P.O. Box 1892
Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu

1 Introduction

Querying is the fundamental mechanism for extracting information from a database. Besides the basic task of query answering, i.e., evaluating a query over a database, data and knowledge representation systems should support other reasoning services related to querying. One of the most important is query containment, i.e., verifying whether for all databases the answer to a query is a subset of the answer to a second query. Checking containment of queries is crucial in several contexts, such as query optimization, query reformulation, knowledge-base verification, information integration, integrity checking, and cooperative answering [18, 12, 4, 6, 21, 7, 13, 20]. Obviously, query containment is also useful for checking equivalence of queries, i.e., verifying whether for all databases the answer to a query is the same as the answer to another query. The main results on query containment are summarized in [10].

In several data management tasks, such as data integration [17], data warehousing, and mobile computing, the data of interest are only accessible through a given set of views. In other words, we would like to access a database, but we have only information about the data satisfying the views. In such a setting, query processing comes in a different form with respect to the traditional framework. Instead of considering a query over of a given database, one should take into account that the query is now processed by relying only on the information about the data satisfying the views. We call this kind of processing *view-based query processing* [8].

Analogously to the traditional setting, also in the context of view-based query processing, the need arises of checking containment between queries. In this context, containment of queries should be determined relative to the set of views, as already noted in the literature [19]. Such form of containment, called *view-based query containment*, is the subject of this paper.

To the best of our knowledge, the first paper dealing with view-based query containment is [19], where the problem, called "relative containment", is studied for variants of conjunctive queries and views.

* This extended abstract reports work that has been published in the Proceedings of the 22nd ACM Symposium on Principles of Database Systems (PODS 2003).

Although previous work on view-based query containment considered only the case of queries expressed in the alphabet of the database (base alphabet), we show here that the problem comes in various forms, depending on whether each of the two queries is expressed over the base alphabet, or the alphabet of the view names (view alphabet). The first contribution of this paper is a thorough analysis of view-based query containment. In particular, we discuss several semantics of containment for all the possible forms, and we study their mutual relationships. The second contribution is a study of view-based query containment in two specific contexts: conjunctive queries in relational databases, and regular path queries in semistructured databases [5, 14, 2]. For both conjunctive and two-way regular path queries and views [8], we provide techniques and complexity bounds for the different variants of view-based query containment. In establishing the results for 2RPQs, we make use of the known connection between view-based query answering and constraint satisfaction [9, 11].

2 Preliminaries

We consider databases and view extensions as finite relational structures. Let Θ be an alphabet of relation symbols, each with an associated arity. A *finite relational structure* (or simply *structure*) $\mathcal{R}$ over Θ is a pair $(\Delta^{\mathcal{R}}, \cdot^{\mathcal{R}})$, where $\Delta^{\mathcal{R}}$ is a finite domain and $\cdot^{\mathcal{R}}$ is a function that assigns to each relation symbol in $r \in \Theta$ a relation $r^{\mathcal{R}}$, also denoted by $r(\mathcal{R})$, of the appropriate arity over $\Delta^{\mathcal{R}}$. Given a *query* Q^{Θ} over Θ, we denote by $Q^{\Theta}(\mathcal{R})$ the result of evaluating Q^{Θ} over $\mathcal{R}$. Note that we explicitly annotate queries with the alphabet over which they are defined. Given two structures $\mathcal{R}_1$ and $\mathcal{R}_2$ over Θ, we use $\mathcal{R}_1 \subseteq \mathcal{R}_2$ to denote that $r(\mathcal{R}_1) \subseteq r(\mathcal{R}_2)$, for each $r \in \Theta$. A query Q^{Θ} is *monotone* if $Q^{\Theta}(\mathcal{R}_1) \subseteq Q^{\Theta}(\mathcal{R}_2)$ whenever $\mathcal{R}_1 \subseteq \mathcal{R}_2$.

Let Σ be a finite alphabet of relation symbols, fixed once and for all, called *base alphabet*. A *database* is a structure over Σ. Consider a database that is accessible only through a collection of views $\mathcal{V}$, and suppose we want to answer a query over the database only on the basis of our knowledge on the views. Specifically, the collection of views is represented by a finite set $\mathcal{V}$ of *view symbols*, each denoting a relation. Each view symbol $V \in \mathcal{V}$ has an associated *view definition* V^{Σ}, which is a query over Σ. A *$\mathcal{V}$-extension* $\mathcal{E}$ is a structure over $\mathcal{V}$.

We consider views to be *sound* [3, 15], i.e., we model a situation where the extension of the views provides a subset of the results of applying the view definitions to the database. Formally, given a set of views $\mathcal{V}$ and a database $\mathcal{B}$, we use $\mathcal{V}^{\Sigma}(\mathcal{B})$ to denote the $\mathcal{V}$-extension $\mathcal{E}$ such that $V(\mathcal{E}) = V^{\Sigma}(\mathcal{B})$, for each $V \in \mathcal{V}$. We say that a $\mathcal{V}$-extension $\mathcal{E}$ is *sound wrt a database* $\mathcal{B}$ if $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(\mathcal{B})$. In other words, for a view $V \in \mathcal{V}$, all the tuples in $V(\mathcal{E})$ must appear in $V^{\Sigma}(\mathcal{B})$, but $V^{\Sigma}(\mathcal{B})$ may contain tuples not in $V(\mathcal{E})$. A set of views $\mathcal{V}$ is *non-constraining* if each $\mathcal{V}$-extension is sound wrt some database, i.e., for each $\mathcal{V}$-extension $\mathcal{E}$ there exists a database $\mathcal{B}$ such that $\mathcal{E} \subseteq \mathcal{V}^{\Sigma}(\mathcal{B})$. Intuitively, all extensions of non-constraining views are admissible, since they are sound wrt some database.

Given a set of views $\mathcal{V}$, a $\mathcal{V}$-extension $\mathcal{E}$, and a query Q^Σ, the set of *certain answers* to Q^Σ wrt $\mathcal{V}$ and $\mathcal{E}$ is the set of tuples t of objects such that $t \in Q^\Sigma(\mathcal{B})$, for every database $\mathcal{B}$ that is consistent with $\mathcal{V}$ and $\mathcal{E}$, i.e., such that $\mathcal{E} \subseteq \mathcal{V}^\Sigma(\mathcal{B})$. *View-based query answering* consists in deciding whether a given tuple of objects is a certain answer to Q wrt $\mathcal{V}$ and $\mathcal{E}$. Given a set of views $\mathcal{V}$ and a query Q^Σ, we denote by $cert_{Q^\Sigma,\mathcal{V}}$ the query that, for every $\mathcal{V}$-extension $\mathcal{E}$, returns the set of certain answers to Q^Σ wrt $\mathcal{V}$ and $\mathcal{E}$.

We consider the case of conjunctive queries over relational databases, in particular, standard conjunctive queries without equalities and without constants. Such queries are monotone and that every set of such views is non-constraining.

We also consider the case of regular path queries over semistructured databases. A *semistructured database* is a finite graph whose nodes represent objects and whose edges are labeled by elements from Σ [5, 1]. An edge from node x to node y labeled by r represents the fact that relation r holds between the object x and the object y. Note that a semistructured database can be viewed as a structure $\mathcal{B}$ over the set Σ of binary relational symbols.

A *regular-path query* (RPQ) over an alphabet Θ of binary relation symbols is expressed as a regular expression or a finite-state automaton over Θ. When evaluated on a structure $\mathcal{R}$ over Θ, an RPQ computes the set of pairs of objects connected in $\mathcal{R}$ by a path in the regular language defined by the RPQ. We consider *two-way regular-path queries* (2RPQs) [8], which extend RPQs with the *inverse* operator. Formally, let $\Theta^\pm = \Theta \cup \{r^- \mid r \in \Theta\}$ be the alphabet including a new symbol r^- for each r in Θ. Intuitively, r^- denotes the inverse of the binary relation r. If $q \in \Theta^\pm$, then we use q^- to mean the *inverse* of q, i.e., if q is r, then q^- is r^-, and if q is r^-, then q^- is r. 2RPQs are expressed by means of regular expressions or finite-state automata over $\Theta^\pm$, whose language is different from the language consisting only of the empty word ε. When evaluated on a structure $\mathcal{R}$ over Θ, a 2RPQ Q^Θ computes the set $Q^\Theta(\mathcal{R})$ of pairs of objects connected in $\mathcal{R}$ by a semipath that conforms to the regular language $L(Q^\Theta)$. A *semipath* in $\mathcal{R}$ from x to y (labeled with $q_1 \cdots q_n$) is a sequence of the form $(y_0, q_1, y_1, \ldots, y_{n-1}, q_n, y_n)$, where $n \geq 0$, $y_0 = x$, $y_n = y$, and for each y_{i-1}, q_i, y_i, we have that $q_i \in \Theta^\pm$, and, if $q_i = r$ then $(y_{i-1}, y_i) \in r(\mathcal{R})$, and if $q_i = r^-$ then $(y_i, y_{i-1}) \in r(\mathcal{R})$. We say that a semipath $(y_0, q_1, \ldots, q_n, y_n)$ *conforms to* Q^Θ if $q_1 \cdots q_n \in L(Q^\Theta)$. Observe that 2RPQs (resp., RPQs) are monotone and that every set of 2RPQ (resp., RPQ) views is non-constraining.

Given two queries Q_1^Θ and Q_2^Θ over the same alphabet Θ, Q_1^Θ is *contained in* Q_2^Θ, denoted $Q_1^\Theta \subseteq Q_2^\Theta$, if for every structure $\mathcal{R}$ over Θ, we have that $Q_1^\Theta(\mathcal{R}) \subseteq Q_2^\Theta(\mathcal{R})$. In this paper we consider query containment relative to a set of views. In particular, we study the notion of *view-based query containment of a query* $Q_1^{\Theta_1}$ *in a query* $Q_2^{\Theta_2}$ *wrt a set of views* $\mathcal{V}$, denoted by $Q_1^{\Theta_1} \subseteq_\mathcal{V} Q_2^{\Theta_2}$, where each Θ_i is either Σ or $\mathcal{V}$. The semantics of this notion depends on Θ_1 and Θ_2, i.e., the alphabets over which the two queries are expressed, and will be discussed for the various cases in the next sections. The complexity results we provide refer to two specific settings: conjunctive queries, where both queries and views are conjunctive, and 2RPQs, where both queries and views are 2RPQs.

3 View-based containment of Q_1^Σ in Q_2^Σ

The first problem we address is view-based containment between two queries over the base alphabet. This is the problem originally studied in [19].

Definition 1. *$Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$ if for every database $\mathcal{B}$, and for every $\mathcal{V}$-extension $\mathcal{E}$ with $\mathcal{E} \subseteq \mathcal{V}(\mathcal{B})$, we have $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{E}) \subseteq cert_{Q_2^\Sigma,\mathcal{V}}(\mathcal{E})$.*

Observe that in this definition we consider only $\mathcal{V}$-extensions that are sound wrt some database. It turns out that, w.l.o.g., we can drop this requirement.

Proposition 1. *$Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$ if and only if for every $\mathcal{V}$-extension $\mathcal{E}$, we have $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{E}) \subseteq cert_{Q_2^\Sigma,\mathcal{V}}(\mathcal{E})$.*

Note that the condition in the proposition above is actually the definition of relative containment in [19]. Intuitively, the condition in the proposition suffices, since for each extension $\mathcal{E}$ that is not contained in $\mathcal{V}(\mathcal{B})$ for some database $\mathcal{B}$, $cert_{Q_i\Sigma,\mathcal{V}}(\mathcal{E})$ becomes trivial and returns all possible tuples of objects in $\mathcal{E}$.

One could also compare the two queries only wrt extensions that correspond exactly to the evaluation of the views over some database. Formally, this notion corresponds to checking whether, for all databases $\mathcal{B}$, we have $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq cert_{Q_2^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B}))$. It turns out that such a notion is equivalent to view-based query containment.

Proposition 2. *$cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq cert_{Q_2^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B}))$ for every database $\mathcal{B}$, if and only if $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$.*

The computational complexity characterization of view-based query containment of Q_1^Σ in Q_2^Σ for conjunctive queries has been provided in [19].

Theorem 1 ([19]). *Checking $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$ is Π_2^P-complete for conjunctive queries.*

For 2RPQs, by exploiting the characterization of $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$, in terms of constraint satisfaction [11], we get the following result.

Theorem 2. *Checking $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\Sigma$ is NEXPTIME-complete for 2RPQs and RPQs.*

4 View-based containment of $Q_1^\mathcal{V}$ in Q_2^Σ

We now study view-based containment of a query $Q_1^\mathcal{V}$ expressed on the view alphabet in a query Q_2^Σ expressed on the base alphabet. Intuitively, we want to check whether, for all view extensions that are sound wrt some database, the evaluation of $Q_1^\mathcal{V}$ yields a subset of the certain answers of Q_1^Σ.

Definition 2. *$Q_1^\mathcal{V} \subseteq_\mathcal{V} Q_2^\Sigma$ if for every database $\mathcal{B}$, and for every $\mathcal{V}$-extension $\mathcal{E}$ with $\mathcal{E} \subseteq \mathcal{V}(\mathcal{B})$, we have $Q_1^\mathcal{V}(\mathcal{E}) \subseteq cert_{Q_2^\Sigma,\mathcal{V}}(\mathcal{E})$.*

Again, the above definition considers only $\mathcal{V}$-extensions that are sound wrt some database. However, as for the case of previous section, it turns out that we can drop this requirement w.l.o.g.

Proposition 3. *$Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ if and only if for every $\mathcal{V}$-extension $\mathcal{E}$ we have $Q_1^{\mathcal{V}}(\mathcal{E}) \subseteq cert_{Q_2^{\Sigma},\mathcal{V}}(\mathcal{E})$.*

As for the case of Section 3, one could also conceive a different notion of containment of $Q_1^{\mathcal{V}}$ in Q_2^{Σ}, namely: for every database $\mathcal{B}$, $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$ is a subset of $cert_{Q_2^{\Sigma},\mathcal{V}}(\mathcal{V}(\mathcal{B}))$. Generally speaking this notion may differ from the one underlying our definition of view-based containment. However it turns out that, if $Q_1^{\mathcal{V}}$ is monotone, the two notions are equivalent.

Proposition 4. *Let $Q_1^{\mathcal{V}}$ be monotone. Then, $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ if and only if for all databases $\mathcal{B}$, we have $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq cert_{Q_2^{\Sigma},\mathcal{V}}(\mathcal{V}(\mathcal{B}))$.*

Interestingly, it turns out that, if both queries $Q_1^{\mathcal{V}}$ and Q_2^{Σ} are monotone, the condition in the proposition above is equivalent to a simpler condition, namely: for every database $\mathcal{B}$ we have that $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\Sigma}(\mathcal{B})$. This is stated in the following proposition.

Proposition 5. *Let $Q_1^{\mathcal{V}}$ and Q_2^{Σ} be monotone. Then, $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ if and only if for every database $\mathcal{B}$, we have $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\Sigma}(\mathcal{B})$.*

Since conjunctive queries are monotone, the above theorem implies that, for conjunctive queries, checking $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ simply amounts to checking whether for every $\mathcal{B}$, $Q_1^{\mathcal{V}}(\mathcal{V}(B)) \subseteq Q_2^{\Sigma}(\mathcal{B})$. This is equivalent to substituting the view symbols appearing in $Q_1^{\mathcal{V}}$ with the corresponding view definitions, thus getting a new query Q_1^{Σ} over the base alphabet Σ, and then checking $Q_1^{\Sigma} \subseteq Q_2^{\Sigma}$.

Theorem 3. *Checking $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ is NP-complete for conjunctive queries.*

Now we turn to the problem of checking whether $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ in the setting of 2RPQs. As for the case of conjunctive queries, since 2RPQs are monotone, checking view-based containment of $Q_1^{\mathcal{V}}$ in Q_2^{Σ} reduces to checking whether for every $\mathcal{B}$, $Q_1^{\mathcal{V}}(\mathcal{V}(B)) \subseteq Q_2^{\Sigma}(\mathcal{B})$. Again, this is equivalent to substituting views with their definitions, thus getting a new query Q_1^{Σ}, and then checking $Q_1^{\Sigma} \subseteq Q_2^{\Sigma}$.

Theorem 4. *Checking $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ is PSPACE-complete for 2RPQs and RPQs.*

5 View-based containment of Q_1^{Σ} in $Q_2^{\mathcal{V}}$

We address the problem of view-based containment between a query over the base alphabet and a query over the view alphabet.

Definition 3. *$Q_1^{\Sigma} \subseteq_{\mathcal{V}} Q_2^{\mathcal{V}}$ if for every database $\mathcal{B}$, and for every $\mathcal{V}$-extension $\mathcal{E} \subseteq \mathcal{V}(\mathcal{B})$, we have $cert_{Q_1^{\Sigma},\mathcal{V}}(\mathcal{E}) \subseteq Q_2^{\mathcal{V}}(\mathcal{E})$.*

Again in this definition we consider only $\mathcal{V}$-extensions that are sound wrt some database. However, we can drop this requirement w.l.o.g. for non-constraining views.

Proposition 6. *Let $\mathcal{V}$ be a set of non-constraining views. Then, $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\mathcal{V}$ if and only if for every $\mathcal{V}$-extension $\mathcal{E}$, we have $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{E}) \subseteq Q_2^\mathcal{V}(\mathcal{E})$*

For conjunctive queries, we know that $cert_{Q_1^\Sigma,\mathcal{V}}$ coincides with the maximally contained rewriting of Q_1^Σ wrt $\mathcal{V}$, which is a possibly exponential union of conjunctive queries, each of linear size in Q_1^Σ [16]. To check non-containment $cert_{Q_1^\Sigma,\mathcal{V}} \not\subseteq Q_2^\mathcal{V}$, we have to guess a conjunctive query in the maximal rewriting of Q_1^Σ, and verify that it is not contained in $Q_2^\mathcal{V}$. Hence we get the following upper bound.

Theorem 5. *Checking $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\mathcal{V}$ is in Π_2^P for conjunctive queries.*

Whether this bound is tight is an open problem.

In the setting of 2RPQs, by exploiting again the characterization in terms of constraint satisfaction [11], we get the following result.

Theorem 6. *Checking $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\mathcal{V}$ is NEXPTIME-complete for 2RPQs and RPQs.*

In fact, the lower bound proof shows that checking $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\mathcal{V}$ is NEXPTIME-complete even for fixed $\mathcal{V}$ and Q_1^Σ.

One could also compare the two queries only wrt extensions that correspond exactly to the evaluation of the views over some database. Formally, this corresponds to checking whether, for every database $\mathcal{B}$, we have that $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B}))$ is a subset of $Q_2^\mathcal{V}(\mathcal{V}(\mathcal{B}))$. Such a notion *differs* from $Q_1^\Sigma \subseteq_\mathcal{V} Q_2^\mathcal{V}$, even for non-constraining views and monotone queries. Indeed, consider $\Sigma = \{R\}$, $\mathcal{V} = \{V_1, V_2\}$ with $V_1^\Sigma = V_2^\Sigma = R$, and queries $Q_1^\Sigma = R$ and $Q_2^\mathcal{V} = V_2$. It is easy to see that, for every database $\mathcal{B}$ we have $cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^\mathcal{V}(\mathcal{V}(\mathcal{B}))$. However, the $\mathcal{V}$-extension $\mathcal{E}$ with $V_1(\mathcal{E}) = \{(c,d)\}$ and $V_2(\mathcal{E}) = \emptyset$ is such that $(c,d) \in cert_{Q_1^\Sigma,\mathcal{V}}(\mathcal{E})$ but $(c,d) \notin Q_2^\mathcal{V}(\mathcal{E})$, thus showing that $Q_1^\Sigma \not\subseteq_\mathcal{V} Q_2^\mathcal{V}$.

Since the two versions of containment differ, the question arises on whether we are able to check containment of Q_1^Σ in $Q_2^\mathcal{V}$ only for those $\mathcal{V}$-extensions that correspond to the evaluation of the view definitions over some database. To this purpose we can exploit the following proposition.

Proposition 7. *Let $Q^\mathcal{V}$ be monotone, and let Q^Σ be the query over Σ obtained by substituting in $Q^\mathcal{V}$ the view symbols with the corresponding view definitions. Then, for every database $\mathcal{B}$, we have $Q^\mathcal{V}(\mathcal{V}(\mathcal{B})) = cert_{Q^\Sigma,\mathcal{V}}(\mathcal{V}(\mathcal{B}))$.*

Observe that, in general, it is not true that, for every $\mathcal{V}$-extension $\mathcal{E}$, we have $Q^\mathcal{V}(\mathcal{E})) = cert_{Q^\Sigma,\mathcal{V}}(\mathcal{E})$. For example, consider $\Sigma = \{R\}$, the set of views $\mathcal{V} = \{V_1, V_2\}$ with $V_1^\Sigma = V_2^\Sigma = R$, and the $\mathcal{V}$-extension $\mathcal{E}$ with $V_1(\mathcal{E}) = \{(a,b)\}$ and $V_2(\mathcal{E}) = \emptyset$. Then, for the query $Q^\mathcal{V} = V_2$, we have that $Q^\mathcal{V}(\mathcal{E}) = \emptyset$, while $cert_{Q^\Sigma,\mathcal{V}}(\mathcal{E}) = \{(a,b)\}$.

By Proposition 7, if $Q_2^{\mathcal{V}}$ is monotone, we can check whether for every database $\mathcal{B}$ we have that $cert_{Q_1^{\Sigma},\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$, by checking whether for every database $\mathcal{B}$ we have that $cert_{Q_1^{\Sigma},\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq cert_{Q_2^{\Sigma}\mathcal{V}}(\mathcal{V}(\mathcal{B}))$, where Q_1^{Σ} is the query obtained from $Q_1^{\mathcal{V}}$ by expanding view symbols with their definitions. By Proposition 2, this can be done by checking whether $Q_1^{\Sigma} \subseteq_{\mathcal{V}} Q_2^{\Sigma}$ (cf. Section 3), and therefore the problem is Π_2^P-complete for conjunctive queries, and NEXPTIME-complete for 2RPQs and RPQs.

6 View-based containment of $Q_1^{\mathcal{V}}$ in $Q_2^{\mathcal{V}}$

The last form of view-based query containment is when the two queries are over the view alphabet.

Definition 4. *$Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\mathcal{V}}$ if for every database $\mathcal{B}$, and for every $\mathcal{V}$-extension $\mathcal{E} \subseteq \mathcal{V}(\mathcal{B})$, we have $Q_1^{\mathcal{V}}(\mathcal{E}) \subseteq Q_2^{\mathcal{V}}(\mathcal{E})$.*

Analogously to the previous case, for non-constraining views, we can drop w.l.o.g. the requirement to consider only $\mathcal{V}$-extensions that are sound wrt some database. Therefore, if views are non-constraining, checking view-based containment reduces to checking containment of the two queries over the view alphabet.

Theorem 7. *Checking $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\mathcal{V}}$ is NP-complete for conjunctive queries, and PSPACE-complete for 2RPQs and RPQs.*

It is interesting to observe that, even for non-constraining views and monotone queries, checking $Q_1^{\mathcal{V}} \subseteq_{\mathcal{V}} Q_2^{\mathcal{V}}$ is *not* equivalent to checking whether, for every database $\mathcal{B}$, we have that $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$. Indeed, consider $\Sigma = \{R\}$, $\mathcal{V} = \{V_1, V_2\}$, with ${V_1}^{\Sigma} = R$ and ${V_2}^{\Sigma} = R$, and queries $Q_1^{\mathcal{V}} = V_1$ and $Q_2^{\mathcal{V}} = V_2$. It is easy to see that for every database $\mathcal{B}$, we have $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$. However, the $\mathcal{V}$-extension $\mathcal{E}$ with $V_1(\mathcal{E}) = \{(a, b)\}$ and $V_2(\mathcal{E}) = \emptyset$ shows that $Q_1^{\mathcal{V}} \not\subseteq_{\mathcal{V}} Q_2^{\mathcal{V}}$.

Since the two versions of containment differ, the question arises on whether we are able to check the condition that for each database $\mathcal{B}$, $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$. It turns out that this is equivalent to substituting view symbols with their definitions in both queries, thus getting new queries Q_1^{Σ} and Q_2^{Σ}, and then checking $Q_1^{\Sigma} \subseteq Q_2^{\Sigma}$. It follows that checking whether, for each database $\mathcal{B}$, $Q_1^{\mathcal{V}}(\mathcal{V}(\mathcal{B})) \subseteq Q_2^{\mathcal{V}}(\mathcal{V}(\mathcal{B}))$, is NP-complete for conjunctive queries, and PSPACE-complete for 2RPQs and RPQs.

7 Conclusions

We have presented a thorough analysis of view-based query containment, showing that the problem comes in various forms, depending on whether each of the two queries is expressed over the base alphabet or the alphabet of the view names. The results presented here also allow us to carry out an analysis of properties of rewritings of queries. In particular, they allow to check for perfectness (vs. exactness) of a rewriting [9, 11].

References

1. S. Abiteboul. Querying semi-structured data. In *Proc. of ICDT'97*, pages 1–18, 1997.
2. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: from Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.
3. S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, pages 254–265, 1998.
4. S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of ACM SIGMOD*, pages 137–148, 1996.
5. P. Buneman. Semistructured data. In *Proc. of PODS'97*, pages 117–121, 1997.
6. P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization technique for unstructured data. In *Proc. of ACM SIGMOD*, pages 505–516, 1996.
7. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR'98*, pages 2–13, 1998.
8. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of PODS 2000*, pages 58–66, 2000.
9. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of LICS 2000*, pages 361–371, 2000.
10. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query answering and query containment over semistructured data. In G. Ghelli and G. Grahne, editors, *Revised Papers of the 8th International Workshop on Database Programming Languages (DBPL 2001)*, volume 2397 of *LNCS*, pages 40–61. Springer, 2002.
11. D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query containment. In *Proc. of PODS 2003*, 2003.
12. S. Chaudhuri, S. Krishnamurthy, S. Potarnianos, and K. Shim. Optimizing queries with materialized views. In *Proc. of ICDE'95*, 1995.
13. M. F. Fernandez, D. Florescu, A. Levy, and D. Suciu. Verifying integrity constraints on web-sites. In *Proc. of IJCAI'99*, pages 614–619, 1999.
14. D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World-Wide Web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.
15. G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 332–347. Springer, 1999.
16. A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
17. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, pages 233–246, 2002.
18. A. Y. Levy and Y. Sagiv. Semantic query optimization in Datalog programs. In *Proc. of PODS'95*, pages 163–173, 1995.
19. T. D. Millstein, A. Y. Levy, and M. Friedman. Query containment for data integration systems. In *Proc. of PODS 2000*, pages 67–75, 2000.
20. T. Milo and D. Suciu. Index structures for path expressions. In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 277–295. Springer, 1999.
21. A. Motro. Panorama: A database system that annotates its answers to queries with their properties. *J. of Intelligent Information Systems*, 7(1), 1996.