What is Query Rewriting?
(Position Paper)

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it

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

Abstract

View-based query processing requires to answer a query posed to a database only on the basis of the information on a set of views, which are again queries over the same database. This problem is relevant in many aspects of database management, and has been addressed by means of two basic approaches, namely, query rewriting and query answering. In the former approach, one tries to compute a rewriting of the query in terms of the views, whereas in the latter, one aims at directly answering the query based on the view extensions. Based on recent results, we first show that already for very simple query languages, a rewriting is in general a co-NP function wrt to the size of view extensions. Hence, the problem arises of characterizing which instances of the problem admit a rewriting that is PTIME. However, a tight connection between view-based query answering and constraint-satisfaction problems, allows us to show that the above characterization is going to be difficult.

1 Introduction

Several recent papers in the literature show that the problem of view-based query processing [25, 2] is relevant in many aspects of database management, including including query optimization, data warehousing, data integration, and query answering with incomplete information. Informally speaking, the problem requires to answer a query posed to a database only on the basis of the information on a set of views, which are again queries over the same database. In query optimization, the problem is relevant because using the views may speed up query processing. In data integration, the views represent the only information sources accessible to answer a query. A data warehouse can be seen as a set of materialized views, and, therefore, query processing reduces to view-based query answering. Finally, since the views provide partial knowledge on the database, view-based query processing can be seen as a special case query answering with incomplete information.

There are two approaches to view-based query processing, called *query rewriting* and *query answering*, respectively. In the former approach, we are given a query Q and a set of view definitions, and the goal is to reformulate the query into an expression, the *rewriting*, that refers only to the views, and provides the answer to Q. Typically, the rewriting is formulated in the same language used for the query and the views. In the latter approach, besides Q and the view definitions, we are also given the extensions of the views. The goal is to compute the set of tuples that are implied by these extensions, i.e., the set of tuples that are in the answer set of Q in all the databases that are consistent with the views.

Notice the difference between the two approaches. In query rewriting, query processing is divided in two steps, where the first re-expresses the query in terms of a given query language over the alphabet of the view names, and the second evaluates the rewriting over the view extensions. In query answering, we do not pose any limit to query processing, and the only goal is to exploit all possible information, in particular the view extensions, to compute the answer to the query.

In the last years a large number of results have been reported for both problems. View-based query rewriting and query answering have been studied under different assumptions on the form of the queries and views. For query rewriting see, e.g., [19, 20, 23, 24, 10, 3, 14, 5, 9], and for query answering see, e.g., [2, 13, 6, 7, 4].

In spite of the large amount of work on the subject, the relationship between view-based query rewriting and view-based query answering is not completely clarified yet. In this paper we focus on this relationship. Abstracting from the language used to express the rewriting, thus generalizing the notion of rewriting considered in the literature, we define a *rewriting* of a query with respect to a set of views as a function that, given the extensions of the views, returns a set of tuples that is contained in the answer set of the query in every database consistent with the views. We call the rewriting that returns precisely such set the *perfect* rewriting of the query wrt the views. Observe that, by evaluating the perfect rewriting over given view extensions, one obtains the same set of tuples provided by view-based query answering. Hence, the perfect rewriting is the best rewriting that one can obtain, given the available information on both the definitions and the extensions of the views.

An immediate consequence of the relationship between perfect rewriting and query answering is that the data complexity of evaluating the perfect rewriting over the view extensions is the same as the data complexity of answering queries using views. Typically, one is interested in queries that can be evaluated in PTIME (i.e., are PTIME functions in data complexity), and hence we would like rewritings to be PTIME as well. For queries and views that are conjunctive queries (without union), the perfect rewriting is a union of conjunctive queries and hence is PTIME [2]. By exploiting the results in [6, 8], we show that already for very simple query languages containing union the perfect rewriting is not PTIME in general. Hence, for such languages it would be interesting to characterize which instances of query rewriting admit a perfect rewriting that is PTIME. However, by establishing a tight connection between view-based query answering and constraint-satisfaction problems (CSP), we show that this is going to be difficult, since it would amount to solve a longstanding open problem for CSP [16, 11].

2 View-based query processing

We consider a simple setting in which a database is constituted by a set of binary relations, and hence can be viewed as an edge-labeled graph. We use as query language *unions of path queries* (UPQs), defined as follows:

$$\begin{array}{lcl} Q & \longrightarrow & P \mid Q_1 \cup Q_2 \\ P & \longrightarrow & R \mid P_1 \circ P_2 \end{array}$$

where R denotes a (binary) database relation, P denotes a *path query*, which is a chaining of database relations, and Q denotes a union of path queries. Observe that such a language is a simplified form both of unions of conjunctive queries [25] and of regular path queries [1].

Let us introduce the problem of view-based query answering [2, 13, 18, 6]. Consider a database that is accessible only through a set $\mathcal{V} = \{V_1, \ldots, V_k\}$ of views, and suppose we want to answer an UPQ only on the basis of our knowledge on the views. Specifically, associated to each view V_i we have:

- its definition $\mathit{def}(V_i)$ in terms of an UPQ over the alphabet Σ;
- information about its extension in terms of a set $\mathit{ext}(V_i)$ of pairs of objects[1].

We denote $(\mathit{def}(V_1), \ldots, \mathit{def}(V_k))$ by $\mathit{def}(\mathcal{V})$, $(\mathit{ext}(V_1), \ldots, \mathit{ext}(V_k))$ by $\mathit{ext}(\mathcal{V})$, and the set of objects appearing in $\mathit{ext}(\mathcal{V})$ by $\mathcal{D}_\mathcal{V}$.

We say that a database DB is *consistent with the views* $\mathcal{V}$ if $\mathit{ext}(V_i) \subseteq \mathit{ans}(\mathit{def}(V_i), DB)$, for each $V_i \in \mathcal{V}$[2]. The *certain answer set of* Q *wrt the views* $\mathcal{V}$ is the set $\mathit{cert}(Q, \mathcal{V}) \subseteq \mathcal{D}_\mathcal{V} \times \mathcal{D}_\mathcal{V}$ such that $(c, d) \in \mathit{cert}(Q, \mathcal{V})$ if and only if $(c, d) \in \mathit{ans}(Q, DB)$, for every DB that is consistent with $\mathcal{V}$.

The problem of *view-based query answering* is the following: Given

- a set $\mathcal{V}$ of views, their definitions $\mathit{def}(\mathcal{V})$, and extensions $\mathit{ext}(\mathcal{V})$,
- a query Q,
- a pair of objects $c, d \in \mathcal{D}_\mathcal{V}$,

decide whether $(c, d) \in \mathit{cert}(Q, \mathcal{V})$.

The complexity of the problem can be measured in three different ways [26]:

- *Data complexity*: as a function of the size of $\mathit{ext}(\mathcal{V})$.
- *Expression complexity*: as a function of the size of Q and of the expressions in $\mathit{def}(\mathcal{V})$.

[1] We assume that objects are represented by constants, and we adopt the *unique name assumption* [21], i.e., different constants denote different objects and therefore different nodes.

[2] This correspond to assume that the views are *sound*. Other assumptions are also possible. For a discussion, see [6].

- *Combined complexity*: as a function of the size of $ext(\mathcal{V})$, Q, and $def(\mathcal{V})$.

Here we focus on data complexity only.

The following theorem characterizes the data complexity of view-based query answering for UPQs.

Theorem 2.1 ([6]) *View-based query answering for UPQs is co-NP-complete in data complexity.*

Proof. The upper bound follows directly from the upper bound for regular path queries shown in [6]. For the lower bound, it is easy to see that the proof in [6] for regular path queries does not exploit reflexive transitive closure, and hence holds also for UPQs. □

The definition of view-based query answering given above reflects two implicit assumptions. (i) The views are *sound*, i.e., from the fact that a pair (a, b) is in $ext(V_i)$ we can conclude that (a, b) is in $ans(def(V_i), DB)$, but not vice-versa. (ii) The domain is *open*, i.e., a database consistent with the views may contain additional objects that do not appear in the view extensions. Other assumptions about the accurateness of the knowledge on the objects of the database and the pairs satisfying the views, have been studied [2, 13, 6].

We now study the relationship between view-based query answering and query rewriting. An instance of *query rewriting* is given by a query Q and a set $\mathcal{V}$ of views with definitions $def(\mathcal{V})$. One then tries to generate a new query Q' over the symbols in $\mathcal{V}$ such that Q' approximates the answer to Q, when V_i is interpreted as $ext(V_i)$, for each $V_i \in \mathcal{V}$. Formally, we require $ans(Q', ext(\mathcal{V})) \subseteq cert(Q, \mathcal{V})$. In the context of UPQs, Q and $def(V_1), \ldots, def(V_k)$ are UPQs over the alphabet Σ, while Q' is an UPQ over the alphabet $\mathcal{V}$.

From a more abstract point of view, we can define a *rewriting of Q wrt $\mathcal{V}$* as a function that, given $ext(\mathcal{V})$, returns a set of pairs of objects that is contained in the certain answer set $cert(Q, \mathcal{V})$. We call the rewriting that returns exactly $cert(Q, \mathcal{V})$ the *perfect rewriting* of Q wrt $\mathcal{V}$. The problem of *view-based query rewriting* is the one of computing a rewriting of Q wrt $\mathcal{V}$. The problem comes in different forms, depending of the properties that we require for the rewriting. In particular:

- It is sometimes interesting to consider rewritings that are expressible in a certain query language, e.g., Datalog.

- It is also interesting to consider rewritings belonging to a certain data complexity class, for example, polynomial time. A rewriting f belongs to a data complexity class $\mathcal{C}$ if the problem of deciding whether a pair of objects (c,d) is in $f(ext(\mathcal{V}))$ is in the class $\mathcal{C}$, where the complexity of the problem is measured with respect to the size of $ext(\mathcal{V})$.

- Finally, it is worth computing rewritings that are maximal in a certain class. A rewriting f of Q wrt $\mathcal{V}$ is *maximal in a class* $\mathcal{C}$ if, for every rewriting $g \in \mathcal{C}$ of Q wrt $\mathcal{V}$, we have that $g(ext(\mathcal{V})) \subseteq f(ext(\mathcal{V}))$ for every $ext(\mathcal{V})$.

An algorithm for view-based query answering is an algorithm that takes as input a query, a set of view definitions, and a set of view extensions, and determines whether a given pair of objects is in the answer set of the query for every database that is consistent with the views. Hence, if we fix the query and the view definitions, we can consider every algorithm for view-based query answering as an algorithm that computes whether a given pair of objects is in the perfect rewriting. This observation establishes a tight connection between view-based query answering [2] and query rewriting [25].

Now, considering that in the present setting view-based query answering is co-NP-complete in data complexity (see Theorem 2.1), we obtain the following result.

Theorem 2.2 *The perfect rewriting of an UPQ wrt UPQ views is a co-NP function. There is an UPQ Q and a set $\mathcal{V}$ of UPQ views such that the rewriting of Q wrt $\mathcal{V}$ is a co-NP-complete function.*

Typically, one is interested in queries that are PTIME functions. Hence, we would like rewritings to be PTIME as well. Unfortunately, even for such a simple language (containing union) as UPQs by Theorem 2.2, the perfect rewritings are not PTIME in general. Hence it would be interesting to characterize which instances of query rewriting admit a perfect rewriting that is PTIME. Note, however, that finding such instances corresponds to finding those instances of view-based query answering that are PTIME in data complexity. Next we show that this is going to be difficult, by exhibiting a tight connection between view-based query answering and constraint satisfaction.

3 Constraint-satisfaction problems

A *constraint-satisfaction problem (CSP)* is traditionally defined in terms of a set of variables, a set of values, and a set of constraints, and asks whether there is an assignment of the variables with the values that satisfies the constraints. An elegant characterization of CSP can be given in terms of homomorphisms between relational structures [11].

A *vocabulary* is a set $V = \{R_1, \ldots, R_t\}$ of predicates, each with an associated arity. A *relational structure* $A = (\Delta^A, \cdot^A)$ over V is a *domain* Δ^A together with an *interpretation function* $\cdot^A$ that assigns to each predicate R_i a relation R_i^A of the appropriate arity over Δ^A. A *homomorphism* $h : A \to B$ between two relational structures A and B over the same vocabulary is a mapping $h : \Delta^A \to \Delta^B$ such that, if $(c_1, \ldots, c_n) \in R^A$, then $(h(c_1), \ldots, h(c_n)) \in R^B$, for every predicate R in the vocabulary.

Let $\mathcal{A}$ and $\mathcal{B}$ be two classes of finite relational structures. The *(uniform) constraint-satisfaction problem* $\mathrm{CSP}(\mathcal{A}, \mathcal{B})$ is the following decision problem: given a structure $A \in \mathcal{A}$ and a structure $B \in \mathcal{B}$ over the same vocabulary, is there a homomorphism $h : A \to B$? We denote such instance as $\mathrm{CSP}(A, B)$, and if such a homomorphism exists we say that $\mathrm{CSP}(A, B)$ is *satisfiable*. We also consider the special case where $\mathcal{B}$ consists of a single relational structure B and $\mathcal{A}$ is the set of all relational structures over the vocabulary of B, and denote it by $\mathrm{CSP}(B)$. Such problem is a (special case of) *non-uniform* constraint-satisfaction problem, i.e., with B fixed, the input is just a structure $A \in \mathcal{A}$. In the case where we take the relational structures to be (directed) graphs, CSP corresponds to *directed-graph homomorphism*. Since general CSP is polynomially equivalent to directed-graph homomorphism [11], that is, for each structure B there is a directed graph G_B such that $\mathrm{CSP}(B)$ is polynomially equivalent to $\mathrm{CSP}(G_B)$, we restrict attention without loss of generality to CSP over directed graphs, unless explicitly stated otherwise.

From the very definition of CSP it follows directly that every $\mathrm{CSP}(\mathcal{A}, \mathcal{B})$ problem is in NP. In general, the complexity of a non-uniform constraint-satisfaction problem $\mathrm{CSP}(B)$ depends on B. For example, $\mathrm{CSP}(K_2)$, is the *Two-Colorability Problem*, while $\mathrm{CSP}(K_3)$ is the *Three-Colorability Problem* (K_n is the n-node complete graph); the former is in PTIME, while the latter is NP-complete. In some cases, e.g., when the domain of B has at most two elements or when B is an undirected graph, it is known that $\mathrm{CSP}(B)$ is either in PTIME or NP-complete [22, 15]. The Dichotomy Conjecture states that this holds for every structure B [11]. (Recall that if PTIME is different than NP then there are problems that are neither in PTIME nor NP-complete [17].) It is an open problem whether the Dichotomy Conjecture holds. A related open question is that of characterizing the structures B for which CSP(B) is in PTIME [11].

4 CSP and view-based query answering

We establish a tight relationship between constraint-satisfaction problems and view-based query answering. We show first that every CSP is polynomially reducible to view-based query answering.

Theorem 4.1 *Let B be a directed graph. There exists an UPQ Q and UPQ views $\mathcal{V}$ with definitions $def(\mathcal{V})$ such that the following holds: for every directed graph A, there are extensions $ext(\mathcal{V})$ and objects c, d such that $(c,d) \notin cert(Q,\mathcal{V})$ if and only if $CSP(A,B)$ is satisfiable.*

Proof (sketch). Let $A = (N_A, E_A)$ and $B = (N_B, E_B)$. We define an instance of view-based query answering as follows:

- The alphabet is $\Sigma = \Sigma_N \cup \Sigma_E$, where $\Sigma_N = \{S_x \mid x \in N_B\} \cup \{F_x \mid x \in N_B\}$ and $\Sigma_E = \{R_{x,y} \mid (x,y) \in E_B\}$.
- The set of objects in the view extensions is $\mathcal{D}_{\mathcal{V}} = N_A \cup \{c,d\}$, where c, d are two symbols not in N_A.
- The views are V_s, V_f, and V_A with

$$
\begin{array}{lll}
def(V_s) = \bigcup_{x \in N_B} S_x & & ext(V_s) = \{(c,a) \mid a \in N_A\} \\
def(V_f) = \bigcup_{x \in N_B} F_x & & ext(V_f) = \{(a,d) \mid a \in N_A\} \\
def(V_A) = \bigcup_{(x,y) \in E_B} R_{x,y} & & ext(V_A) = E_A
\end{array}
$$

  Intuitively, the extension of V_A represents A, while V_s and V_f are used to connect c and d to all nodes of A, using respectively the "start" relations S_x and "final" relations F_x.

- The query is

$$
\begin{array}{rlll}
Q = & \displaystyle\bigcup_{x,y\in N_B,\, x\neq y} & S_x \circ F_y & \cup \\
& \displaystyle\bigcup_{\substack{x\in N_B \\ y\neq x,\, (y,z)\in E_B}} & S_x \circ R_{y,z} \circ F_z & \cup \\
& \displaystyle\bigcup_{\substack{x\in N_B \\ (x,y)\in E_B,\, z\in N_B\setminus\{y\}}} & S_x \circ R_{x,y} \circ F_z &
\end{array}
$$

It is possible to show that there is a homomorphism from A to B if and only if $(c,d) \notin cert(Q,\mathcal{V})$. □

The reduction in the proof of Theorem 4.1 is polynomial, so we get the following corollary.

Corollary 4.2 *Every CSP is polynomially reducible to view-based query answering.*

Theorem 4.1 exhibits a very strong connection between CSP and view-based query answering. In particular, since in the reduction, the query and the view definitions depend only on graph B, and only the view extensions depend on graph A, the theorem shows that non-uniform CSP can be polynomially reduced to query rewriting. As a consequence, if we had a method to decide whether an instance of query rewriting admits a perfect rewriting that is PTIME, we would then be able to precisely characterize those instances of non-uniform CSP that are in PTIME. As discussed in [16, 11], this is a longstanding open problem that appears to be difficult to solve.

5 Conclusions

We have set up a framework that clarifies the relationships between view-based query rewriting and view-based query answering. Based on such a framework, we have first shown that the perfect rewriting is in general a co-NP function wrt to the size of view extensions. We have then turned our attention to the problem of characterizing which instances of query rewriting admit a rewriting that is PTIME. Based on a tight connection between view-based query answering and constraint-satisfaction problems, we have shown that the above characterization is going to be difficult.

The discussion above shows that in general there is a tradeoff between completeness of a rewriting and the efficiency of using the rewriting to compute the answers to the query. To retain efficiency, one has in general to give up the perfectness of the rewriting, and adopt weaker notions of completeness. For example, one can fix a priori the language of the rewriting (such language should have PTIME data complexity) and find the maximal rewritings expressible in such a language. In particular, if the maximal rewriting is logically equivalent to the query (*exact rewriting*) and all views are exact, then the rewriting is also perfect. However, such a rewriting may not exist.

References

[1] S. Abiteboul. Querying semi-structured data. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, pages 1–18, 1997.

[2] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[3] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, 2000. To appear.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'99)*, pages 194–204, 1999.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 389–398, 2000.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2000)*, 2000. To appear.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Sym. on Logic in Computer Science (LICS 2000)*, 2000. To appear.

[9] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART*

Sym. on Principles of Database Systems (PODS'99), pages 155–166, 1999.

[10] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997.

[11] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction. *SIAM J. on Computing*, 28:57–104, 1999.

[12] M. L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, Los Altos, 1987.

[13] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer-Verlag, 1999.

[14] J. Gryz. Query folding with inclusion dependencies. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 126–133, 1998.

[15] P. Hell and J. Nešetřil. On the complexity of H-coloring. *J. of Combinatorial Theory, Series B*, 48:92–110, 1990.

[16] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 205–213, 1998.

[17] R. E. Ladner. On the structure of polynomial time reducibility. *J. of the ACM*, 22:155–171, 1975.

[18] A. Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 402–412, 1996.

[19] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[20] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering queries using templates with binding patterns. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'95)*, 1995.

[21] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 119–140. Plenum Publ. Co., New York, 1978. Republished in [12].

[22] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. of the 10th ACM Sym. on Theory of Computing (STOC'78)*, pages 216–226, 1978.

[23] D. Srivastava, S. Dar, H. V. Jagadish, and A. Levy. Answering queries with aggregation using views. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 318–329, 1996.

[24] O. G. Tsatalos, M. H. Solomon, and Y. E. Ioannidis. The GMAP: A versatile tool for phyisical data independence. *Very Large Database J.*, 5(2):101–118, 1996.

[25] J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, 1997.

[26] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Sym. on Theory of Computing (STOC'82)*, pages 137–146, 1982.