Bounded Regular Path Queries in View-Based Data Integration*

Gösta Grahne[1] and Alex Thomo[2]

[1] Concordia University, Montreal, Canada, grahne@cs.concordia.ca
[2] University of Victoria, Victoria, Canada, thomo@cs.uvic.ca

Abstract. In this paper we study the problem of deciding boundedness of (recursive) regular path queries over views in data integration systems, that is, whether a query can be re-expressed without recursion. This problem becomes challenging when the views contain recursion, thereby potentially making recursion in the query uncessary. We define and solve two related problems of boundedness of regular path queries. One of the problems asks for the existence of a bound, and the other, more restricted one, asks if the query is bounded within a given parameter. For the more restricted version we show it PSPACE complete, and obtain a constructive method for optimizing the queries. For the existential version of boundedness, we show it PTIME reducible to the notorious problem of limitedness in distance automata. This problem has received a lot attention in the formal language community, but only exponential time algorithms are currently known.

1 Introduction

The compile-time query optimization is one of the key factors for the enormous success of database systems today. Notably, the majority of the influential work on query optimizers dealt with SQL queries, for which the negation free ones correspond to datalog queries without recursion.

Nevertheless, in the research community from the mid 1980's to the mid 1990's, another theme was the study of (recursive) datalog (see *e. g.* [27]). Unfortunately, most decision problems related to query optimization turned out to be undecidable. One of the undecidable problems was the boundedness of datalog, which is to decide whether a given recursive datalog query is equivalent to one without recursion. The importance of this is that should we be able to re-express a recursive datalog query as another query without recursion, then we could use the optimization machinery for non-recursive queries, which over the years has been proven to be very efficient and successful in commercial systems.

Notably, a well-behaved fragment of datalog, which is both natural and quite general, did emerge in the mid 1990's, in the context of semistructured data (graph data) (see [27]). This fragment is the class of regular queries, whose basic element is that of regular path queries.

The semi-structured data model [1] is now widely used as a foundation for reasoning about a multitude of applications, where the data is best formalized in terms of labeled graphs. Such data is usually found in Web information systems, XML data repositories, digital libraries, communication networks, and so on.

Regarding the query languages for semi-structured data, virtually all of them provide the possibility for the user to query the database through regular expressions. The boundedness problem for regular path queries is decidable. Simply, one has to build a finite automaton and check whether there is a cycle on a path between an initial and a final state.

This simplicity is not true anymore when the regular path queries are on an alphabet where the symbols represent views, which can in turn can have recursive definitions. Such queries are prominent today in information integration systems, where the data-sources are represented as a

* A preliminary version of this paper appeared at IDEAS'07.

set of views over a *global schema*. The data is described by the *local schema*, which in the semi-structured context is the set of view names. By using the view definitions, the user query (expressed on the global schema) is rewritten by the integration engine in terms of the local schema. Finally, the obtained view-based rewriting is used to extract the answer from the data on the local schema. This is commonly referred in the literature as the *local-as-view* (LAV) approach for data integration (see [18]).

Thus, in practice, we have to deal with machine generated regular path queries (rewritings) on view names, and the above-mentioned simple check for recursion is not sufficient.

A classical example given in [3] is the following. Suppose that we have a single view $V = R^*$ and the view-based rewriting is V^*. Clearly, this is equivalent with just V, which is more efficient than V^* to be answered on the data source.

We note here that, the view-based rewritings, generated by the method proposed in [3] always contain *all* the recursion possible, as long as the containment of the rewriting to the original query is preserved. The problem of "minimizing" a rewriting was first proposed in that same paper ([3]) but it has been open since then.

In general, the problem is more complicated than the example above, which only illustrates a case where the non-recursive rewriting reduces to a single-letter word. As a more elaborated example, consider the view $V = R^+$. Now, for any given (natural) number k, take the query $Q = R^*.R^k$. The rewriting computed by [3] and [8] will be $(V^k)^+$, and clearly we need to have a way to infer that in fact the only word needed from this language is the word V^k, which has length k.

In this paper, by solving the boundedness problem for regular path queries over views, we show that for the *exact* view-based assumption in data integration, it is sometimes possible to replace a view-based rewriting with a not necessarily (purely) algebraically equivalent non-recursive one, without loosing any answers. The exact view assumption is usually the implicit assumption in a multitude of applications such as datawarehouses and enterprise data integration applications (see e.g. [9]), and has received considerable attention in the research community (see e.g. [6, 2]).

Furthermore, we obtain an optimal algorithm, which takes as input a view-based expression and a number k, and returns an equivalent expression without recursion (if such exists), in which the length of the longest word does not exceed k.

Depending on the application, we might be interested only in the *existence* of the above number k. Namely, we would like to know, for a given expression (query) on the view definitions, whether there exists a number k, such that the sub-language of the words of length not more than k is equivalent with the language captured by the original expression. Clearly, this amounts to deciding whether a query can be equivalently re-expressed without recursion. We show that our existential problem is polynomial time reducible to the intricate limitedness problem for distance automata, intensely investigated by Hashiguchi and others [11, 13, 14, 19, 25, 23].

The rest of the paper is organized as follows. In Section 2, we give the necessary background on regular path queries and views in LAV data integration. In Section 3, we present a characterization theorem for the regular path query equivalence with respect to a set of views. In the same section, we define two notions of boundedness, the k-boundedness and finite boundedness. In Section 4, we present our results for the k-boundedness. In Section 5, we solve the finite boundedness problem by giving a reduction to the limitedness problem in distance automata. Finally, Section 6 concludes the paper.

2 Basic Definitions

We consider a database to be an edge-labeled graph. This graph model is typical in semistructured data, where the nodes of the database graph represent the objects and the edges represent the attributes of the objects, or relationships between the objects.

Formally, let Δ be a finite alphabet. We shall call Δ the *database alphabet.* Elements of Δ will be denoted $R, S, \ldots$. As usual, Δ^* denotes the set of all finite words over Δ. Words will be denoted by

$u, w, \ldots$. We also assume that we have a universe of objects, and objects will be denoted $a, b, c, \ldots$. A *database* $\mathcal{D}$ over Δ is a is a subset of $N \times \Delta \times N$, where N is a finite set of objects, that we usually will call nodes. We view a database as a directed labeled graph, and interpret a triple (a, R, b) as a directed edge from a to object b, labeled with R. If there is a path labeled $R_1, R_2, \ldots, R_k$ from a node a to a node b we write $a \xrightarrow{R_1 R_2 \ldots R_k} b$.

A *(user) query* Q is a regular language over Δ. For the ease of notation, we will blur the distinction between regular languages and regular expressions that represent them. Let Q be a query and $\mathcal{D}$ a database. Then, the *answer* to Q on $\mathcal{D}$ is defined as

$$ans(Q, \mathcal{D}) = \{(a, b) : a \xrightarrow{w} b \text{ in } \mathcal{D} \text{ for some } w \in Q\}.$$

Let $V_1, \ldots, V_n$ be languages (queries) on alphabet Δ. We will call them *views* and associate with each V_i a view name v_i.

We call the set $\Omega = \{v_1, \ldots, v_n\}$ the *outer alphabet*, or *view alphabet*. For each $v_i \in \Omega$, we set $h(v_i) = V_i$. The substitution h associates with each view name v_i in the Ω alphabet the language V_i. The substitution h is applied to words, languages, and regular expressions in the usual way (see e.g. [15]).

A *view graph* is database $\mathcal{V}$ over Ω. In other words, a view graph is a database where the edges are labeled with symbols from Ω. View graphs can also be queried by regular path queries over Ω. However, as explained below these are not queries given by the user, but rather rewritings computed by the system.

In a LAV ("local-as-view") information integration system [18], we have the "global schema" Δ, the "source schema" Ω, and the "assertion" $h : \Omega \to 2^{\Delta^*}$. The only extensional data available is a view graph $\mathcal{V}$ over Ω (see also [20, 26, 7, 5][3]).

The user queries are expressed on the global schema Δ, and the system has to answer based solely on the information provided by the views. In order to do this, the system has to reason with respect to the set of *possible databases* over Δ that $\mathcal{V}$ could represent. Under the *exact view* assumption, a view graph $\mathcal{V}$ defines a set $poss(\mathcal{V})$ of databases as follows:

$$poss(\mathcal{V}) = \{\mathcal{D} \ : \ \mathcal{V} = \bigcup_{i \in \{1,\ldots,n\}} \{(a, v_i, b) : (a, b) \in ans(V_i, \mathcal{D})\}\}.$$

(Recall that $V_i = h(v_i)$.) The above definition reflects the intuition that a database $\mathcal{D}$ is *possible* with respect to a view graph $\mathcal{V}$ if the computation of the V_i views on $\mathcal{D}$ gives precisely the set of objects (a, b) which when connected by the appropriate v_i edges yield graph $\mathcal{V}$.

The meaning of querying a view graph through the global schema in a LAV information integrations system is defined as follows. Let Q be a query over Δ. Then

$$\text{ANS}(Q, \mathcal{V}) = \bigcap_{\mathcal{D} \in poss(\mathcal{V})} ans(Q, \mathcal{D}).$$

This is also called the *certain answer* of Q with respect to the given views (cf. [18]).

Henceforth, we will consider only view graphs which are *valid*, that is, the view graphs for which the set of possible databases is not empty. Under the exact view assumption, not all view graphs are valid. As an example, consider a single view $V = R^*$, and the view graph $\mathcal{V} = \{(a, v, b), (b, v, c)\}$. It is easy to see that $poss(\mathcal{V}) = \emptyset$. The reason is that $\mathcal{V}$ "misses" a v-edge from a to c.

There are two approaches for computing $\text{ANS}(Q, \mathcal{V})$. The first one is to use an exponential procedure in the size of the data in order to completely compute $\text{ANS}(Q, \mathcal{V})$ (see [4]). There is little that one can better hope for, since in the same paper it has been proven that to decide whether a tuple belongs to $\text{ANS}(Q, \mathcal{V})$ is co-NP complete with respect to the size of data.

[3] Regarding corresponding LAV scenarios for relational data.

The second approach is to first compute a view-based rewriting Q' for Q, as in [3]. For example, if $Q = (R + S)^*(T + U)$ and $V_1 = (R + S)^*$, $V_2 = T$, then the computed rewriting is $Q' = v_1^* v_2$.

Such rewritings are regular path queries on Ω. We can approximate $\text{ANS}(Q, \mathcal{V})$ by $ans(Q', \mathcal{V})$, which can be computed in polynomial time with respect to the size of data. In general, for a view-based rewriting Q' computed by the algorithm of [3], we have that

$$ans(Q', \mathcal{V}) \subseteq \text{ANS}(Q, \mathcal{V}),$$

with equality when the rewriting is exact ([4]). In the rest of the paper, we will assume that the data-integration system follows the second approach.

3 Query Equivalence and Boundedness

Consider two queries (rewritings), Q_1 and Q_2 over an alphabet $\Sigma \in \{\Delta, \Omega\}$. We say that a query Q_1 is Σ-*contained* in a query Q_2 denoted $Q_1 \subseteq_\Sigma Q_2$ iff the answer to Q_1 is contained to the answer to Q_2, on all databases over Σ. We say that Q_1 is Σ-*equivalent* to Q_2 and write $Q_1 \equiv_\Sigma Q_2$, when $Q_1 \subseteq_\Sigma Q_2$ and $Q_2 \subseteq_\Sigma Q_1$. It is easy to see that the above query containment coincides with the (regular) language containment of Q_1 and Q_2, and that the query equivalence coincides with the language equality, i.e. $Q_1 \subseteq_\Sigma Q_2$ iff $Q_1 \subseteq Q_2$ and $Q_1 \equiv_\Sigma Q_2$ iff $Q_1 = Q_2$.

Let Q_1 and Q_2 be queries over Ω. We say that Q_1 is Ω/Δ-*contained* in Q_2, denoted $Q_1 \subseteq_{\Omega/\Delta} Q_2$, iff $h(Q_1) \subseteq_\Delta h(Q_2)$. Likewise, Q_1 is Ω/Δ-*equivalent* to Q_2 denoted $Q_1 \equiv_{\Omega/\Delta} Q_2$, when $Q_1 \subseteq_{\Omega/\Delta} Q_2$ and $Q_2 \subseteq_{\Omega/\Delta} Q_1$. It is easy to see that Ω-containment $Q_1 \subseteq_\Omega Q_2$, implies Ω/Δ-containment $Q_1 \subseteq_{\Omega/\Delta} Q_2$ but not vice-versa. As an example, if $Q_1 = v$, $Q_2 = v^*$ (where $v \in \Omega$), and $h(v) = R^*$, then Q_1 is Ω/Δ-equivalent with Q_2, although they are not Ω-equivalent.

We now have the following theorem.

Theorem 1. *Let Q_1 and Q_2 be queries over Ω. Under the exact view assumption, $Q_1 \subseteq_{\Omega/\Delta} Q_2$ iff for each valid view graph $\mathcal{V}$ over Ω, $ans(Q_1, \mathcal{V}) \subseteq ans(Q_2, \mathcal{V})$.*

PROOF. "If." Assume that $Q_1 \not\subseteq_{\Omega/\Delta} Q_2$. Then, there exists a word $w = R_1 \dots R_m$, such that $w \in h(Q_1)$, but $w \notin h(Q_2)$. Let $a, b, c_1, \dots, c_{m-1}$ be objects from the universe of objects. We construct the database $\mathcal{D} = \{(a, R_1, c_1), \dots, (c_{m-1}, R_m, b)\}$. From $\mathcal{D}$ we construct a view graph $\mathcal{V}$, by computing $ans(V_i, \mathcal{D})$, for all $i \in [1, n]$.

The view graph $\mathcal{V}$ is not empty since it is computed on the database $\mathcal{D}$, which has a path that spells w, which in turn is in $h(Q_1) \subseteq h(\Omega^*)$. So, there exist a word on Ω, whose "Δ-expansion" (applying h on it) contains w. Clearly, this Ω-word has to be spelled by some path in $\mathcal{V}$. Furthermore, $\mathcal{V}$ is valid since $\mathcal{D} \in poss(\mathcal{V})$.

It is easy to verify that $(a, b) \in ans(Q_1, \mathcal{V})$ but $(a, b) \notin ans(Q_2, \mathcal{V})$ (which is a contradiction). To see this, let us assume that $(a, b) \in ans(Q_2, \mathcal{V})$, i.e. there exists a word $v_{i_1} \dots v_{i_k} \in Q_2$, which is spelled by a path connecting a with b in $\mathcal{V}$. By the construction of $\mathcal{V}$, this means that $w = R_1 \dots R_m \in h(v_{i_1} \dots v_{i_k})$, and since $h(v_{i_1} \dots v_{i_k}) \subseteq h(Q_2)$, we get that $w \in h(Q_2)$, which is a contradiction.

"Only if." Assume that there exists a valid view graph $\mathcal{V}$ for which $ans(Q_1, \mathcal{V}) \not\subseteq ans(Q_2, \mathcal{V})$. Then, there exists a tuple $(a, b) \in ans(Q_1, \mathcal{V})$ and $(a, b) \notin ans(Q_2, \mathcal{V})$. Since $(a, b) \in ans(Q_1, \mathcal{V})$, there is a path from a to b in $\mathcal{V}$ spelling a word (of Q_1), say $v_{i_1} \dots v_{i_k}$ (on Ω). Now, let $\mathcal{D}$ be an arbitrary database in $poss(\mathcal{V})$. Since $\mathcal{D}$ is a possible database with respect to $\mathcal{V}$, the objects a and b will be in $\mathcal{D}$, and furthermore there will be a Δ-path from a to b spelling a word, say $w_{i_1} \dots w_{i_k}$, where $w_{i_1} \in h(v_{i_1})$, $\dots$, $w_{i_k} \in h(v_{i_k})$. From the fact that $Q_1 \equiv_{\Omega/\Delta} Q_2$, we have $\{v_{i_1} \dots v_{i_k}\} \subseteq_{\Omega/\Delta} Q_2$, which implies that the word $w_{i_1} \dots w_{i_k} \in h(Q_2)$. Thus, there exists a word $v_{j_1} \dots v_{j_m} \in Q_2$, such that $w_{i_1} \dots w_{i_k} \in h(v_{j_1} \dots v_{j_m})$. This means that, we can find a word $w_{j_1} \dots w_{j_m}$, where $w_{j_1} \in h(v_{j_1})$, $\dots$, $w_{j_m} \in h(v_{j_m})$, and such that $w_{j_1} \dots w_{j_m} = w_{i_1} \dots w_{i_k}$. This amounts to saying that, the path spelling $w_{i_1} \dots w_{i_k}$ in $\mathcal{D}$, can be seen to spell $w_{j_1} \dots w_{j_m}$ as well, i.e. there exist objects $c_1, \dots, c_{m-1}$

in $\mathcal{D}$, and the (on focus) path can written as $aw_{j_1}c_1 \ldots c_{m-1}w_{j_m}b$. Clearly, $(a, c_1) \in ans(h(v_{j_1}), \mathcal{D})$, $\ldots$, $(c_{m-1}, b) \in ans(h(v_{j_m}), \mathcal{D})$.

Finally, because of the exactness assumption for the views, we have that $(a, v_{j_1}, c_1) \in \mathcal{V}$, $\ldots$, $(c_{m-1}, v_{j_m}, b) \in \mathcal{V}$, which implies in turn that $(a, b) \in ans(\{v_{j_1} \ldots v_{j_m}\}, \mathcal{V})$, i.e. $(a, b) \in ans(Q_2, \mathcal{V})$, which is a contradiction. □

Corollary 1. *Let Q_1 and Q_2 be queries over Ω. Under the exact view assumption, $Q_1 \equiv_{\Omega/\Delta} Q_2$ iff for each valid view graph $\mathcal{V}$ over Ω, $ans(Q_1, \mathcal{V}) = ans(Q_2, \mathcal{V})$.*

The importance of the above corollary and theorem is that it allows us to minimize as much as possible a query on Ω (i.e. a view-based rewriting) without loosing query-power as long as we preserve Ω/Δ-equivalence, which is algebraically weaker than Ω-equivalence.

The above does not hold when we drop the exactness assumption for the views and consider them sound only.[4] As an example, consider a view V, which is Δ-equivalent with V^*, and a view graph $\mathcal{V} = \{(a, v, b), (b, v, c)\}$. For this $\mathcal{V}$, we have that $ans(v^*, \mathcal{V}) \neq ans(v, \mathcal{V})$. Clearly, the answer of V will be equal to the answer of V^* on each database on Δ, but because the view is assumed to be sound we cannot enforce $\mathcal{V}$ to have an additional v-edge from a to c.

Now, let us denote with $Q^{(k)}$ the set of *all* words in Q, which have length of not more than k. Obviously, $Q^{(0)} \subseteq Q^{(1)} \subseteq \ldots \subseteq Q^{(k)} \subseteq \ldots \subseteq Q$.

In the following, we formally define two boundedness problems.

Problem 1. k-Boundedness
Instance. A query Q on Ω, a set of views $\mathbf{V} = \{V_1, \ldots, V_n\}$ on Δ, and a $k \in \mathbb{N}$.
Answer. "Yes" if and only if $Q^{(k)} \equiv_{\Omega/\Delta} Q$.

If the answer to this problem is "Yes," we say that the query Q is *k-bounded*.

Problem 2. Finite Boundedness
Instance. A query Q on Ω, and a set of views $\mathbf{V} = \{V_1, \ldots, V_n\}$ on Δ.
Answer. "Yes" if and only if there exists $k \in \mathbb{N}$ such that Q is k-bounded.

If the answer to this problem is "Yes," we say that the query Q is *finitely bounded*.

4 *k*-Boundedness

As a first observation, the problem of k-boundedness is decidable. For this,

1. Construct an automaton (NFA) for Q.
2. Construct an automaton for Ω^k. This automaton will have $k+1$ states, $\{s_0, s_1 \ldots, s_k\}$ (all of them final and s_0 initial) and transitions (s_i, v, s_{i+1}), for each $v \in \Omega$, and $0 \leq i \leq k-1$.
3. Compute the intersection $I = Q \cap \Omega^k$ by constructing the Cartesian product of the above two automata.
4. Check $I \equiv_{\Omega/\Delta} Q$ by checking the regular language equivalence $h(I) \equiv_{\Delta} h(Q)$.

Language Ω^k is the set of all words (on Ω) of length not more than k. Constructing an automaton for Ω^k is pseudo-polynomial in k (if a binary representation for k is assumed).[5] Intersecting Q with Ω^k extracts all the words of Q having a length not more than k. This gives $Q^{(k)}$, i.e. $I = Q^{(k)}$. Step 4 is a regular language equivalence check which can be done in PSPACE.

We turn now on the lower bound for deciding the k-boundedness.

[4] Under the *sound view* assumption, we have

$$poss(\mathcal{V}) = \{\mathcal{D} \ : \ \mathcal{V} \subseteq \bigcup_{i \in \{1,\ldots,n\}} \{(a, v_i, b) : (a, b) \in ans(V_i, \mathcal{D})\}\}.$$

[5] Constructing Ω^k can be done in $O(k)$ time, but since k is assumed to be represented in binary (having a $\log_2 k$ size), this construction takes in fact a time which is exponential in the representation size of k.

Theorem 2. *The problem of deciding k-boundedness is PSPACE-hard.*

PROOF. We will reduce the NFA universality problem to the k-boundedness problem. The universality problem says: given an NFA $\mathcal{A}$, is $\Delta^* \subseteq L(\mathcal{A})$? The universality problem is PSPACE-complete [16]. Without loss of generality, we can restrict the problem to the class of NFA's which accept the empty word ϵ, and all the single-letter words on Δ, as well.

Let $\mathcal{A}$ be an arbitrary NFA on Δ. We consider a single view $V = L(\mathcal{A})$, and the corresponding view alphabet $\Omega = \{v\}$. We show that $\Delta^* \subseteq L(\mathcal{A})$ iff v^* is 1-bounded.

The only if direction is obvious. For the if direction, suppose $v \equiv_{\Omega/\Delta} v^*$. From this, we have that $V = V \cdot V$. Now, using also the fact that $\Delta \subset V$, we have that

$$V \cdot \Delta \subseteq V \cdot V = V, \quad V \cdot \Delta^2 \subseteq V \cdot \Delta = V, \quad \ldots, \quad V \cdot \Delta^n \subseteq V \cdot \Delta = V, \quad \ldots$$

from which we can conclude that $V \cdot \Delta^* \subseteq V$. Since we assumed that $\epsilon \in L(A) = V$, we get $\{\epsilon\} \cdot \Delta^* \subseteq V$, which is $\Delta^* \subseteq V$, thereby proving our claim. □

If parameter k is fixed, then from all the above, we have

Corollary 1 *The problem of k-boundedness is PSPACE-complete.*

5 Finite Boundedness

5.1 Upper bound

In this section, we will polynomially reduce the finite boundedness problem to the problem of limitedness in distance automata. As a consequence, we obtain an exponential time upper bound for the finite boundedness problem.

A *distance automaton* $\mathscr{A}$ consists of a finite set of states P, an alphabet Δ, a starting state s, a set of final states F, and a transition relation $\tau \subseteq P \times \Delta \times \{0, 1\} \times P$. (Observe that a distance automaton is ϵ-free.) Intuitively, a distance automaton is an automaton with transitions weighted by 1 or 0.

Let p and q be two states of a distance automaton $\mathscr{A}$ (as above), and let π be a path between them, spelling a word w. Note that there can be more than one path from p to q spelling w. Let

$$d_{\mathscr{A}}(p, w, q) = \mathit{inf}\{\mathit{weight}(\pi) : \pi \text{ is a path spelling } w, \text{ from } p \text{ to } q \text{ in } \mathscr{A}\},$$

where the weight of a path is the sum of the weights of the edges along it.

Also, for a state p and a subset of states $M \subseteq P$, we define

$$d_{\mathscr{A}}(p, w, M) = \mathit{inf}\{d_{\mathscr{A}}(p, w, q) : q \in M\}.$$

Let $\mathcal{A}$ be the (classical) automaton obtained from $\mathscr{A}$ by removing the transition weights. We define $L(\mathscr{A})$ to be $L(\mathcal{A})$. Now, the *distance* of $\mathscr{A}$ is defined as

$$d(\mathscr{A}) = \mathit{sup}\{d_{\mathscr{A}}(s, w, F) : w \in L(\mathscr{A})\}.$$

We say that a distance automaton $\mathscr{A}$ is *limited* if $d(\mathscr{A}) < \infty$.

Our reduction. Given a query Q, and a set of view definitions $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$, we shall construct a distance automaton $\mathscr{A}_{Q,\mathbf{V}}$, such that Q is finitely bounded w.r.t. $\mathbf{V}$ if and only if $\mathscr{A}_{Q,\mathbf{V}}$ is limited in distance.

First, we construct in polynomial time an ϵ-free automaton (NFA) $\mathcal{A}_Q$ for Q. Let this automaton be $\mathcal{A}_Q = (P, \Omega, \tau, s, F)$ consisting of a finite set of states P, alphabet Ω, a starting state s, a set of final states F, and a transition relation $\tau \subseteq P \times \Delta \times P$.

Also for each view V_i, where $1 \leq i \leq n$, we construct a zero-weighted distance automaton $\mathscr{A}_{V_i} = (P_i, \Delta, \tau_i, s_i, F_i)$ which is obtained from an ϵ-free automaton (NFA) for V_i by "weighting" the transitions with 0.

Now, from the query automaton $\mathcal{A}_Q$ and the view automata we construct a distance automaton $\mathscr{A}_{Q,\mathbf{V}}$ on alphabet Δ. Namely, using the states of $\mathcal{A}_Q$ to initialize, the construction of $\mathscr{A}_{Q,\mathbf{V}}$ proceeds as follows: For each transition (p, v_i, q) in $\mathcal{A}_Q$:

1. create a copy of distance automaton $\mathscr{A}_{V_i}$ which for simplicity is denoted with the same signature $(P_i, \Delta, \tau_i, s_i, F_i)$
2. add the following set of transitions
   (a) $\{(p, R, 0, r) : (s_i, R, 0, r) \in \tau_i\}$
   (b) $\{(r, R, 1, q) : (r, R, 0, t) \in \tau_i \text{ and } t \in F_i\}$
   (c) $\{(p, R, 1, q) : (s_i, R, 0, r) \in \tau_i \text{ and } r \in F_i\}$.

Transitions in the first set are for "jumping" from state p to all the states of (copy) automaton $\mathscr{A}_{V_i}$, that are reachable in one step from the initial state s_i of $\mathscr{A}_{V_i}$. These transitions are weighted by 0. Once the "control" is transferred to $\mathscr{A}_{V_i}$, the only way to "escape" from it and go to state q is by scanning a word in language V_i. Scanning of such words (except their last symbol) is done at zero cost through the transitions of $\mathscr{A}_{V_i}$. Their last symbol causes a "jump" to state q and it is marked by "paying" a price of one unit. This is handled by the transitions of the second set. Finally, transitions of the third set are for handling the special case of single symbol words accepted by $\mathscr{A}_{V_i}$.

Further, we set s as the initial state and F as the set of final states in $\mathscr{A}$. From the above descriptions, it is clear that

Proposition 1. *$L(\mathscr{A}_{Q,\mathbf{V}}) = h(Q)$.*

We now show that

Theorem 3. *Q is finitely bounded with respect to V if and only if $\mathscr{A}_{Q,\mathbf{V}}$ is limited.*

Proof.

"If." Since $\mathscr{A}_{Q,\mathbf{V}}$ is limited, there exists a $k \in \mathbb{N}$ such that $d(\mathscr{A}_{Q,\mathbf{V}}) \leq k$. This means that for each word w in $h(Q) = L(\mathscr{A}_{Q,\mathbf{V}})$ there exists a path π of transitions going from initial state s to some final state f and weighted by some integer $j \leq k$. By the construction of $\mathscr{A}_{Q,\mathbf{V}}$, path π "pays" a unit cost each time that it spells some word in a view language. Thus, there exist a word $v_{i_1} \dots v_{i_j} \in Q$, such that $w \in h(v_{i_1} \dots v_{i_j})$. Since the length of $v_{i_1} \dots v_{i_j}$ is not more than k, we have that $v_{i_1} \dots v_{i_j} \in Q^{(k)}$. Hence, $h(Q) \subseteq h(Q^{(k)})$, which immediately implies that $Q^{(k)} \equiv_{\Omega/\Delta} Q$, i.e. Q is k-bounded.

"Only if." Since Q is finitely bounded, for each word w in $h(Q) = L(\mathscr{A}_{Q,\mathbf{V}})$ there exists a word $v_{i_1} \dots v_{i_j} \in Q$, where $j \leq k$, such that $w \in h(v_{i_1} \dots v_{i_j})$. By the construction of $\mathscr{A}_{Q,\mathbf{V}}$ we have that $d_{\mathscr{A}_{Q,\mathbf{V}}}(s, w, F) \leq j \leq k$, where s is the initial state and F is the set of final states in $\mathscr{A}_{Q,\mathbf{V}}$. Since, w was an arbitrary word in $h(Q) = L(\mathscr{A}_{Q,\mathbf{V}})$, we finally get $d(\mathscr{A}_{Q,\mathbf{V}}) \leq k$. □

Hence, the finite boundedness is reducible to the limitedness of distance automata. Since such an automaton is constructible in polynomial time, we have that the reduction is polynomial as well.

The first solution to the limitedness of distance automata was obtained by Hashiguchi in [11]. By now, it is known that the problem is PSPACE-complete (see [19] and [17] for the lower and upper bound, respectively).

5.2 Lower bound

Regarding the lower complexity bound, it can be shown that the well-known problem of finite power property (FPP) for regular languages can be reduced to our (query) finite boundedness problem. FPP asks whether for a given regular language, say L, there exists an $m \in \mathbb{N}$, such that

$$L^* = \{\epsilon\} \cup L \cup L^2 \cup \ldots \cup L^m.$$

Now, this can be reduced to the finite boundedness problem by considering a single view $V = L$, a corresponding alphabet $\Omega = \{v\}$, and a query v^*.

As shown by Weber in [28], the FPP problem is PSPACE-hard. From this, and the PSPACE-completeness of the limitedness problem, we conclude that our boundedness problem is PSPACE-complete.

5.3 Bibliographical Remark

Hashiguchi in [12] defined and solved the following representation problem.

> Let L and $L_1, \ldots, L_n$ be regular languages over some alphabet Δ. Is L expressible in terms of $L_1, \ldots, L_n$ by using only union and concatenation?

To solve this problem, he presented a reduction to the limitedness problem in distance automata. The reduction works by building a distance automaton $\mathscr{B}$ which is exponential in the size of an automaton $\mathcal{A}$ for L. Then, the above problem is answered affirmatively if and only if $L(\mathscr{B}) = L$ and $\mathscr{B}$ is limited in distance. Checking the limitedness of $\mathscr{B}$ (by the most efficient algorithm) can be done in exponential time in the size of $\mathscr{B}$, i.e. in doubly-exponential time in the size of $\mathcal{A}$.

Now, our problem of finite boundedness can be reduced to this representation problem, by setting $L = h(Q)$ and $L_1 = V_1, \ldots, L_n = V_n$. However, this would give us a *doubly-exponential* algorithm for deciding the finite boundedness.

Finally, we note that a reverse reduction, from Hashiguchi's problem to the finite boundedness problem, does not seem possible. Given L and $L_1, \ldots, L_n$, there does not always exist a Q such that $h(Q) = L$. Rather, one can only compute the biggest Q such that $h(Q) \subseteq L$.

6 Conclusions

We have formally defined and solved two problems of boundedness for view-based query rewritings. These problems are related to the problem of minimizing view-based rewritings, which was first proposed by [3].

We believe that the problems we have solved are of significant importance for the optimization of queries in LAV data integration systems. This is because should we be able to express a view-based query rewriting without using recursion, then we could use the optimization machinery for non-recursive queries.

Acknowledgment. We would like to thank Daniel Kirsten for pointing to us the [28] paper. Also, we would like to thank an anonymous reviewer for constructive comments on a previous version of this paper.

References

1. Abiteboul S., P. Buneman and D. Suciu. *Data on the Web : From Relations to Semistructured Data and XML.* Morgan Kaufmann Pulishers. San Francisco, Ca., 1999.
2. Bravo L., and Bertossi L. Deductive databases for computing certain and consistent answers from mediated data integration systems. *J. Applied Logic* 3(1): 329–367, 2005.
3. Calvanese D., G. Giacomo, M. Lenzerini and M. Y. Vardi. Rewriting of Regular Expressions and Regular Path Queries. *Proc. PODS '99*, pp. 194–204.
4. Calvanese D., G. Giacomo, M. Lenzerini and M. Y. Vardi. Answering Regular Path Queries Using Views. *Proc. ICDE '00*, pp. 389–398
5. Deutsch A., Y. Katsis and Y. Papakonstantinou. Determining Source Contribution in Information Integration Systems. *Proc. PODS '05*

6. Flesca, S., and Greco, S. Rewriting queries using views. *IEEE Trans. Knowl. Data Eng.* 13(6): 980–995, 2001
7. Grahne G. and Mendelzon A. O. Tableau Techniques for Querying Information Sources through Global Schemas. *Proc. ICDT '99*, pp. 332–347
8. Grahne G., and A. Thomo. An Optimization Technique for Answering Regular Path Queries *Proc. WebDB '00*, pp. 99–104.
9. Jonson H., and Xiaoyan Q. DB2 information integrator V8.1: Under the Hood. *ARISE '04 http://www.scs.carleton.ca/∼nvillanu/Presentations/IBM.ppt*
10. Hashiguchi K. A Decision Procedure for the Order of Regular Events. *Theoretical Computer Science* 8: 69–72, 1979
11. Hashiguchi K. Limitedness Theorem on Finite Automata with Distance Functions. *J. Comp. Syst. Sci.* 24(2): 233–244, 1982
12. Hashiguchi K. Representation Theorems on Regular Languages. *J. Comput. Syst. Sci.* 27(1): 1983, pp. 101–115.
13. Hashiguchi K. Improved Limitedness Theorems on Finite Automata with Distance Functions. *Theoretical Computer Science* 72(1): 27–38, 1990
14. Hashiguchi K. New upper bounds to the limitedness of distance automata. *Theoretical Computer Science* 233(1-2): 19–32, 2000
15. Hopcroft J. E., and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley. Reading Ma., 1979.
16. Hunt H. B. III, D. J. Rosenkrantz, and T. G. Szymanski, On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Languages. *J. Comp. Syst. Sci.* 12 (2) : 222–268, 1976
17. Kirsten D. Distance desert automata and the star height problem. *R.A.I.R.O. - Informatique Theorique et Applications* 39 (3) : 455–509, 2005.
18. Lenzerini M. Data Integration: A Theoretical Perspective. *Proc. PODS '02*, pp. 233–246
19. Leung H. Limitedness Theorem on Finite Automata with Distance Functions: An Algebraic Proof. *Theoretical Computer Science* 81 (1) : 137–145, 1991
20. Levy A. Y., Mendelzon A. O., Sagiv Y., Srivastava D. Answering Queries Using Views. *Proc. PODS '95*, pp. 95-104
21. Mendelzon A. O., and P. T. Wood, Finding Regular Simple Paths in Graph Databases. *SIAM J. Comp.* 24 (6) : 1235–1258, 1995.
22. Mendelzon A. O. G. A. Mihaila and T. Milo. Querying the World Wide Web. *Int. J. Dig. Lib.* 1 (1) : 57–67, 1997
23. Pin. J. E. Tropical Semirings, *in Idempotency*, J. Gunawardena (ed.) Cambridge University Press, pp. 50–69, 1998
24. Simon. I. Limited Subsets of a Free Monoid. *Proc. FOCS '78* , pp. 143–150
25. Simon. I. On Semigroups of Matrices over the Tropical Semiring. *Informatique Theorique et Applications* 28 (3-4) : 277–294, 1994
26. Ullman J. D. Information Integration Using Logical Views. *Proc. ICDT '97*, pp. 19-40.
27. Vardi. M. Y. A Call to Regularity. *Proc. PCK50 - Principles of Computing & Knowledge, Paris C. Kanellakis Memorial Workshop '03*, pp. 11
28. Weber A. Distance Automata Having Large Finite Distance or Finite Ambiguity. *Mathematical Systems Theory* 26 (2) : 169–185, 1993